Package: admiralonco (via r-universe)

August 18, 2024

Type Package

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

Version 1.1.0

Description Programming oncology specific Clinical Data Interchange
Standards Consortium (CDISC) compliant Analysis Data Model
(ADaM) datasets in 'R'. 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>). The
package is an extension package of the 'admiral' package.

Language en-US

License Apache License (>= 2)

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

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

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends R (>= 3.5)

Imports admiral (>= 1.0.0), admiraldev (>= 1.0.0), dplyr (>= 0.8.4),
lifecycle (>= 0.1.0), lubridate (>= 1.7.4), magrittr (>= 1.5),
rlang (>= 0.4.4), tidyselect (>= 1.0.0)

Suggests pharmaversesdtm, pharmaverseadam, devtools, diffdf, lintr,
pkgdown, testthat (>= 3.0.0), knitr, methods, miniUI,
rmarkdown, roxygen2, spelling, stringr, tibble, usethis, covr,
DT, metatools, cli

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website gert

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

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

RemoteRef HEAD

RemoteSha e2ad84912b36327311270c651c2bcf060ced1ea9

Contents

---

admiral_adrs   *Response Analysis Dataset*

Description

An example response analysis dataset

Usage

```
admiral_adrs
```

Format

An object of class tbl_df (inherits from tbl, data.frame) with 3694 rows and 75 columns.

Source

Derived from the adsl, rs and tu datasets using {admiralonco} (https://github.com/pharmaverse/admiralonco/blob/main/inst/templates/ad_adrs.R)

aval_resp	*Map Character Response Values to Numeric Values*

Description

Map character response values like "PR" or "SD" to numeric values.

Usage

```
aval_resp(arg)
```

Arguments

arg	Character vector

Value

- 1 if arg equals "CR",

- 2 if arg equals "PR",

- 3 if arg equals "SD",

- 4 if arg equals "NON-CR/NON-PD",

- 5 if arg equals "PD",

- 6 if arg equals "NE",

- 7 if arg equals "MISSING",

- NA_real_ otherwise

Author(s)

Stefan Bundfuss

Examples

```
aval_resp(c("CR", "PR", "SD", "NON-CR/NON-PD", "PD", "NE", "MISSING", "ND", NA_character_))
```

---

death_event   *Pre-Defined Time-to-Event Source Objects*

Description

These pre-defined `tte_source` objects can be used as input to `admiral::derive_param_tte()`.

Usage

```
death_event

lastalive_censor

pd_event

lasta_censor

rand_censor

trts_censor
```

Details

To see the definition of the various objects simply print the object in the R console, e.g. `print(death_event)`. For details of how to use these objects please refer to `admiral::derive_param_tte()`.

Printing an object will display input dataset_name, filter (if applicable), date variable, and appropriate values for EVNTDESC, CNSDTDSC, SRCDOM, SRCVAR, and SRCSEQ.

See Also

`admiral::derive_param_tte()`, `admiral::tte_source()`, `admiral::event_source()`, `admiral::censor_source()`

Examples

```
# This shows the definition of all pre-defined `tte_source` objects that ship
# with {admiralonco}
for (obj in admiral::list_tte_source_objects(package = "admiralonco")$object) {
  cat(obj, "\n")
  print(get(obj))
  cat("\n")
}
```

| derive_param_bor | *Adds a Parameter for Best Overall Response (without confirmation)* |

Description

[Superseded] The `derive_param_bor()` function has been superseded in favor of `derive_extreme_event()`.

Adds a parameter for best overall response, without confirmation, optionally up to first progressive disease

Usage

```
derive_param_bor(
  dataset,
  dataset_adsl,
  filter_source,
  source_pd = NULL,
  source_datasets = NULL,
  reference_date,
  ref_start_window,
  missing_as_ne = FALSE,
  aval_fun,
  set_values_to,
  subject_keys = get_admiral_option("subject_keys")
)
```

Arguments

`dataset`	The input dataframe from which the Best Overall Response will be derived from and added to.
	The columns `PARAMCD`, `ADT`, and `AVALC` and the columns specified in `subject_keys` and `reference_date` are expected.
	After applying `filter_source` and/or `source_pd` the column `ADT` and the columns specified by `subject_keys` must be a unique key of the dataframe.
	Permitted Values: a `data.frame()` object
`dataset_adsl`	ADSL input dataset.
	The columns specified in the subject_keys argument are expected. For each subject in the passed `dataset` a new row is added to the input `dataset`. Columns in `dataset_adsl` that also appear in `dataset` will be populated with the appropriate subject-specific value for these new rows.
	Permitted Values: a `data.frame()` object
`filter_source`	Filter to be applied to `dataset` to derive the Best Overall Response
`source_pd`	Date of first progressive disease (PD)
	If the parameter is specified, the observations of the input `dataset` for deriving the new parameter are restricted to observations up to the specified date. Observations at the specified date are included. For subjects without first PD date all observations are take into account.

Permitted Values: a date_source object (see date_source() for details)

source_datasets

Source dataframe to be used to calculate the first PD date

A named list of dataframes is expected (although for BOR) only one dataframe is needed. It links the dataset_name from source_pd with an existing dataframe.

For example if source_pd = pd_date with

```
pd_date <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == PD
)
```

and the actual response dataframe in the script is myadrs, source_datasets = list(adrs = myadrs) should be specified.

reference_date   Reference date

The reference date is used along with ref_start_window to determine those records that occur before and after ADT (see Details section for further information). Usually it is treatment start date (TRTSDT) or randomization date (RANDDT).

Permitted Values: a numeric date column

ref_start_window

Stable disease time window

The ref_start_window is used along with reference_date to determine those records that occur before and after ADT (i.e. for a record determine whether ADT >= reference_date + ref_start_window), see Details section for further information.

Permitted Values: a non-negative numeric scalar

missing_as_ne   Consider no assessments as "NE"?

If the argument is set to TRUE, the response is set to "NE" for subjects in dataset_adsl without an assessment in the dataset after the filter has been applied. Otherwise, the response is set to "MISSING" for these subjects.

Permitted Values: a logical scalar

aval_fun   *Deprecated*, please use set_values_to instead.

Function to map character analysis value (AVALC) to numeric analysis value (AVAL)

The (first) argument of the function must expect a character vector and the function must return a numeric vector.

set_values_to   New columns to set

A named list returned by exprs() defining the columns to be set for the new parameter, e.g. exprs(PARAMCD = "BOR", PARAM = "Best Overall Response") is expected. The values must be symbols, character strings, numeric values, or NA.

subject_keys   Columns to uniquely identify a subject

Permitted Values: A list of symbols created using exprs().

Details

Calculates the best overall response (BOR) parameter, as detailed below.

Records after PD can be removed using the source_pd and source_datasets arguments.

Note:

1. All CR, PR and PD response records are considered for Best Overall Response.

2. All SD or NON-CR/NON-PD records where ADT >= reference_date + ref_start_window are also considered for Best Overall Response.

3. Subjects with **ONLY** an SD or NON-CR/NON-PD records where ADT < reference_date + ref_start_window are assigned a Best Overall Response of NE.

4. The Best Response, from the records in steps 1 to 3, is then selected in the following order of preference: CR, PR, SD, NON-CR/NON-PD, PD, NE, MISSING

5. The AVAL column is added and set using the aval_fun(AVALC) function

6. The columns specified by the set_values_to parameter and records are added to the dataframe passed into the dataset argument

Note: Any responses of SD or NON-CR/NON-PD that occur before reference_date + ref_start_window are ignored in the calculation of BOR. All other responses are included in the calculation of BOR, irrespective of the number of days from the reference date.

Also Note: All columns from the input dataset are kept. For subjects with no records in the input dataset (after the filter is applied) all columns are kept from ADSL which are also in the input dataset. Columns which are not to be populated for the new parameter or populated differently (e.g. RSSTRESC, VISIT, PARCATy, ANLzzFL, ...) should be overwritten using the set_values_to parameter.

Value

The dataframe passed in the dataset argument with additional columns and/or rows as set in the set_values_to argument.

Author(s)

Stephen Gormley

See Also

Other superseded: derive_param_clinbenefit(), derive_param_confirmed_bor(), derive_param_confirmed_resp() derive_param_response(), filter_pd()

Examples

```
library(magrittr)
library(dplyr)
library(tibble)
library(lubridate)
library(admiral)

# Create ADSL dataset
```

```
adsl <- tribble(
  ~USUBJID, ~TRTSDTC,
  "1",      "2020-01-01",
  "2",      "2019-12-12",
  "3",      "2019-11-11",
  "4",      "2019-12-30",
  "5",      "2020-01-01",
  "6",      "2020-02-02",
  "7",      "2020-02-02",
  "8",      "2020-04-01"
) %>%
  mutate(
    TRTSDT = ymd(TRTSDTC),
    STUDYID = "XX1234"
  )

# Create ADRS dataset
ovr_obs <- tribble(
  ~USUBJID, ~ADTC, ~AVALC, ~ANL01FL,
  "1", "2020-01-01", "PR", "Y",
  "1", "2020-02-01", "CR", "Y",
  "1", "2020-02-16", "NE", "Y",
  "1", "2020-03-01", "CR", "Y",
  "1", "2020-04-01", "SD", "Y",
  "2", "2020-01-01", "SD", "Y",
  "2", "2020-02-01", "PR", "Y",
  "2", "2020-03-01", "SD", "Y",
  "2", "2020-03-13", "CR", "Y",
  "3", "2019-11-12", "CR", "Y",
  "3", "2019-12-02", "CR", "Y",
  "3", "2020-01-01", "SD", "Y",
  "4", "2020-01-01", "PR", "Y",
  "4", "2020-03-01", "SD", "N",
  "4", "2020-04-01", "SD", "Y",
  "4", "2020-05-01", "PR", "Y",
  "4", "2020-05-15", "NON-CR/NON-PD", "Y",
  "5", "2020-01-01", "PR", "Y",
  "5", "2020-01-10", "SD", "Y",
  "5", "2020-01-20", "PR", "Y",
  "5", "2020-05-15", "NON-CR/NON-PD", "Y",
  "6", "2020-02-06", "PR", "Y",
  "6", "2020-02-16", "CR", "Y",
  "6", "2020-03-30", "PR", "Y",
  "6", "2020-04-12", "PD", "Y",
  "6", "2020-05-01", "CR", "Y",
  "6", "2020-06-01", "CR", "Y",
  "7", "2020-02-06", "PR", "Y",
  "7", "2020-02-16", "CR", "Y",
  "7", "2020-04-01", "NE", "N"
) %>%
  mutate(PARAMCD = "OVR")

pd_obs <-
```

```
    bind_rows(tribble(
      ~USUBJID, ~ADTC,          ~AVALC,
      "2",       "2020-03-01", "Y",
      "4",       "2020-02-01", "Y"
    ) %>%
      mutate(PARAMCD = "PD"))

adrs <- bind_rows(ovr_obs, pd_obs) %>%
  mutate(
    ADT = ymd(ADTC),
    STUDYID = "XX1234"
  ) %>%
  select(-ADTC) %>%
  derive_vars_merged(
    dataset_add = adsl,
    by_vars     = exprs(STUDYID, USUBJID),
    new_vars    = exprs(TRTSDT)
  )

pd_date <- date_source(
  dataset_name = "adrs",
  date         = ADT,
  filter       = PARAMCD == "PD"
)

aval_fun_pass <- function(arg) {
  case_when(
    arg == "CR" ~ 11,
    arg == "PR" ~ 22,
    arg == "SD" ~ 33,
    arg == "NON-CR/NON-PD" ~ 44,
    arg == "PD" ~ 55,
    arg == "NE" ~ 66,
    arg == "MISSING" ~ 77,
    TRUE ~ NA_real_
  )
}

# Derive best overall response parameter
derive_param_bor(
  adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd_date,
  source_datasets = list(adrs = adrs),
  aval_fun = aval_fun_pass,
  reference_date = TRTSDT,
  ref_start_window = 28,
  set_values_to = exprs(
    PARAMCD = "BOR",
    PARAM = "Best Overall Response"
  )
) %>%
```

```
    filter(PARAMCD == "BOR")
```

---

derive_param_clinbenefit

Adds a Parameter for Clinical Benefit

Description

[**Superseded**] The derive_param_clinbenefit() function has been superseded in favor of derive_extreme_event().

Adds a parameter for clinical benefit/disease control

Usage

```
derive_param_clinbenefit(
  dataset,
  dataset_adsl,
  filter_source,
  source_resp,
  source_pd = NULL,
  source_datasets,
  reference_date,
  ref_start_window,
  aval_fun,
  clinben_vals = c("CR", "PR", "SD", "NON-CR/NON-PD"),
  set_values_to,
  subject_keys = get_admiral_option("subject_keys")
)
```

Arguments

dataset   Input dataset. This is the dataset to which the clinical benefit rate parameter will
                 be added.

                 The variables PARAMCD, AVALC, ADT, and those specified by the subject_keys
                 parameter and the reference_date parameter are expected.

                 After applying filter_source and/or source_pd the variable ADT and the vari-
                 ables specified by subject_keys must be a unique key of the dataset.

dataset_adsl   ADSL input dataset.

                 The variables specified for subject_keysis expected. For each subject of the
                 specified dataset a new observation is added to the input dataset. Variables in
                 dataset_adsl that also appear in dataset will be populated with the appropri-
                 ate subject-specific value for these new observations.

filter_source   Filter condition in dataset that represents records for overall disease response
                 assessment for a subject at a given timepoint, e.g. PARAMCD == "OVR" or PARAMCD
                 == "OVRLRESP".

source_resp   A date_source object specifying the dataset, date variable, and filter condition
                 used to identify response status.

source_pd	A `date_source` object specifying the dataset, date variable, and filter condition used to identify disease progression.
source_datasets	
	A named list of data sets is expected.
	The list must contain the names provided by the `dataset_name` field of the `date_source()` objects specified for `source_pd` and `source_resp`.
reference_date	Name of variable representing the index date for `ref_start_window`. A variable providing a date. An unquoted symbol is expected.
ref_start_window	
	Integer representing number of days from `reference_date` that must elapse before an evaluable non-PD assessment counts toward determining clinical benefit.
aval_fun	*Deprecated*, please use `set_values_to` instead.
	Function to map character analysis value (AVALC) to numeric analysis value (AVAL)
	The (first) argument of the function must expect a character vector and the function must return a numeric vector.
clinben_vals	A vector of response values to be considered when determining clinical benefit.
set_values_to	A named list returned by `exprs()` containing new variables and their static value to be populated for the clinical benefit rate parameter records, e.g. `exprs(PARAMCD = "CBR", PARAM = "Clinical Benefit Rate")`.
subject_keys	A named list returned by `exprs()` containing variables used to uniquely identify subjects.

Details

Clinical benefit/disease control is first identified by looking for subjects having response status, and then derived for subjects that have at least one evaluable non-PD response assessment prior to first PD (Progressive Disease) (i.e., responses inclusive of CR, PR, SD, and NON-CR/NON-PD) and after a specified amount of time from a reference date (`ref_start_window`).

Note: The user input values they wish to include when determining clinical benefit using the argument `clinben_vals`. The default values for this are CR, PR, SD, and NON-CR/NON-PD, as listed above. In the below example, eligible values be limited to CR and PR.

Example: `clinben_vals <- c("CR", "PR")`

1. The input dataset (`dataset`) is restricted to the observations matching `filter_source` and to observations before or at the date specified by `source_pd`.

2. This dataset is further restricted to include user-generated response assessments from `clinben_vals` or include response assessments of CR, PR, SD, and NON-CR/NON-PD, exclude missing response assessments, and exclude those less than `ref_start_window` after `reference_date`. The earliest assessment by ADT is then selected.

3. The dataset identified by `dataset` in `source_resp` is restricted according to its `filter` argument. The variable corresponding to the `date` parameter of `source_resp` is considered together with ADT from the previous step.

4. For the observations being added to `dataset`, ADT is set to the earlier of the first assessment date representing an evaluable non-PD assessment prior to first PD, or the date representing the start of response.

5. For the observations being added to dataset, AVALC is set to
   - Y for those subjects in the dataset meeting the criteria for clinical benefit above
   - N for subjects not meeting the clinical benefit criteria in dataset or the dataset identified in source_resp
   - N for subjects present in dataset_adsl but not present in dataset or the dataset identified in source_resp.

6. AVAL is derived using AVALC as input to the function specified in aval_fun.

7. The variables specified by set_values_to are added to the new observations with values equal to the values specified in the same.

8. The new observations are added to dataset. Variables held in common between dataset and dataset_adsl are kept for the new observations, and are populated with their values from dataset_adsl.

Value

The input dataset with a new parameter for clinical benefit

Author(s)

Andrew Smith

See Also

Other superseded: derive_param_bor(), derive_param_confirmed_bor(), derive_param_confirmed_resp(), derive_param_response(), filter_pd()

Examples

```
library(lubridate)
library(dplyr)
library(admiral)

adsl <- tibble::tribble(
  ~USUBJID, ~TRTSDT,
  "01",      ymd("2020-01-14"),
  "02",      ymd("2021-02-16"),
  "03",      ymd("2021-03-09"),
  "04",      ymd("2021-04-21")
) %>%
  mutate(STUDYID = "AB42")

adrs <- tibble::tribble(
  ~USUBJID, ~PARAMCD, ~AVALC, ~ADT,
  "01",      "RSP",     "Y",    ymd("2021-03-14"),
  "02",      "RSP",     "N",    ymd("2021-05-07"),
  "03",      "RSP",     "N",    NA,
  "04",      "RSP",     "N",    NA,
  "01",      "PD",      "N",    NA,
  "02",      "PD",      "Y",    ymd("2021-05-07"),
  "03",      "PD",      "N",    NA,
```

```
  "04",      "PD",      "N",      NA,
  "01",      "OVR",     "SD",     ymd("2020-03-14"),
  "01",      "OVR",     "PR",     ymd("2021-04-13"),
  "02",      "OVR",     "PR",     ymd("2021-04-08"),
  "02",      "OVR",     "PD",     ymd("2021-05-07"),
  "02",      "OVR",     "CR",     ymd("2021-06-20"),
  "03",      "OVR",     "SD",     ymd("2021-03-30"),
  "04",      "OVR",     "NE",     ymd("2021-05-21"),
  "04",      "OVR",     "NA",     ymd("2021-06-30"),
  "04",      "OVR",     "NE",     ymd("2021-07-24"),
  "04",      "OVR",     "ND",     ymd("2021-09-04"),
) %>%
  mutate(STUDYID = "AB42", ANL01FL = "Y") %>%
  derive_vars_merged(
    dataset_add = adsl,
    by_vars = exprs(STUDYID, USUBJID),
    new_vars = exprs(TRTSDT)
  )

pd <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "PD" & AVALC == "Y" & ANL01FL == "Y"
)

resp <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "RSP" & AVALC == "Y" & ANL01FL == "Y"
)

derive_param_clinbenefit(
  dataset = adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
  source_resp = resp,
  source_pd = pd,
  source_datasets = list(adrs = adrs),
  reference_date = TRTSDT,
  ref_start_window = 28,
  set_values_to = exprs(
    PARAMCD = "CBR"
  )
) %>%
  filter(PARAMCD == "CBR")
```

---

derive_param_confirmed_bor

Adds a Parameter for Confirmed Best Overall Response

Description

[**Superseded**] The `derive_param_confirmed_bor()` function has been superseded in favor of
`derive_extreme_event()`.

Adds a parameter for confirmed best overall response (BOR)

Usage

```
derive_param_confirmed_bor(
  dataset,
  dataset_adsl,
  filter_source,
  source_pd = NULL,
  source_datasets = NULL,
  reference_date,
  ref_start_window,
  ref_confirm,
  max_nr_ne = 1,
  accept_sd = FALSE,
  missing_as_ne = FALSE,
  aval_fun,
  set_values_to,
  subject_keys = get_admiral_option("subject_keys")
)
```

Arguments

dataset	Input dataset
	The `PARAMCD`, `ADT`, and `AVALC` variables and the variables specified by `subject_keys` and `reference_date` are expected.
	After applying `filter_source` and/or `source_pd` the variable `ADT` and the variables specified by `subject_keys` must be a unique key of the dataset.
dataset_adsl	ADSL input dataset
	The variables specified for `subject_keys` are expected. For each subject of the specified dataset a new observation is added to the input dataset.
filter_source	Source filter
	All observations in `dataset_source` fulfilling the specified condition are considered for deriving the confirmed best overall response.
source_pd	Date of first progressive disease (PD)
	If the parameter is specified, the observations of the input dataset for deriving the new parameter are restricted to observations up to the specified date. Observations at the specified date are included. For subjects without first PD date all observations are take into account.
	Permitted Values: a `date_source` object (see `admiral::date_source()` for details)
source_datasets	
	Source dataset for the first PD date

A named list of datasets is expected. It links the `dataset_name` from `source_pd` with an existing dataset.

For example if `source_pd = pd_date` with

```
pd_date <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == PD
)
```

and the actual response dataset in the script is `myadrs`, `source_datasets = list(adrs = myadrs)` should be specified.

reference_date   Reference date

The reference date is used for the derivation of `"SD"` and `"NON-CR/NON-PD"` response (see "Details" section). Usually it is treatment start date (`TRTSDT`) or randomization date (`RANDDT`).

Permitted Values: a numeric date variable

ref_start_window

Stable disease time window

Assessments at least the specified number of days after the reference date (i.e. where `ADT >= reference_date + ref_start_window`) with response `"CR"`, `"PR"`, `"SD"`, or `"NON-CR/NON-PD"` are considered for `"SD"` or `"NON-CR/NON-PD"` response.

Permitted Values: a non-negative numeric scalar

ref_confirm   Minimum time period for confirmation

The assessment and the confirmatory assessment for `"CR"` and `"PR"` have to be at least the specified number of days apart.

max_nr_ne   The specified number of `"NE"` assessments between the assessment and the confirmatory assessment for `"CR"` and `"PR"` response is accepted.

Permitted Values: a non-negative numeric scalar

accept_sd   Accept `"SD"` for `"PR"`?

If the argument is set to `TRUE`, one `"SD"` assessment between the assessment and the confirmatory assessment for `"PR"` response is accepted. Otherwise, no `"SD"` assessment must occur between the two assessments.

Permitted Values: a logical scalar

missing_as_ne   Consider no assessments as `"NE"`?

If the argument is set to `TRUE`, the response is set to `"NE"` for subjects without an assessment in the input dataset. Otherwise, the response is set to `"MISSING"` for these subjects.

Permitted Values: a logical scalar

aval_fun   *Deprecated*, please use `set_values_to` instead.

Function to map character analysis value (AVALC) to numeric analysis value (AVAL)

The (first) argument of the function must expect a character vector and the function must return a numeric vector.

set_values_to   Variables to set

A named list returned by exprs() defining the variables to be set for the new parameter, e.g. exprs(PARAMCD = "CBOR", PARAM = "Confirmed Best Overall Response") is expected. The values must be symbols, character strings, numeric values, or NA.

subject_keys   Variables to uniquely identify a subject

A list of symbols created using exprs() is expected.

Details

1. The input dataset (dataset) is restricted to the observations matching filter_source and to observations before or at the date specified by source_pd.

2. The following potential confirmed responses are selected from the restricted input dataset:

   - "CR": An assessment is considered as complete response (CR) if
     - AVALC == "CR",
     - there is a confirmatory assessment with AVALC == "CR" at least ref_confirm days after the assessment,
     - all assessments between the assessment and the confirmatory assessment are "CR" or "NE", and
     - there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment.

   - "PR": An assessment is considered as partial response (PR) if
     - AVALC == "PR",
     - there is a confirmatory assessment with AVALC %in% c("CR", "PR") at least ref_confirm days after the assessment,
     - all assessments between the assessment and the confirmatory assessment are "CR", "PR", "SD", or "NE",
     - there is no "PR" assessment after a "CR" assessment in the confirmation period,
     - there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment, and
     - if the accept_sd argument is set to TRUE, one "SD" assessment in the confirmation period is accepted. Otherwise, no "SD" assessment must occur within the confirmation period.

   - "SD": An assessment is considered as stable disease (SD) if
     - AVALC %in% c("CR", "PR", "SD") and
     - the assessment is at least ref_start_window days after reference_date.

   - "NON-CR/NON-PD": An assessment is considered as NON-CR/NON-PD if
     - AVALC = "NON-CR/NON-PD" and
     - the assessment is at least ref_start_window days after reference_date.

   - "PD": An assessment is considered as progressive disease (PD) if AVALC == "PD".

   - "NE": An assessment is considered as not estimable (NE) if
     - AVALC == "NE" or
     - AVALC %in% c("CR", "PR", "SD", "NON-CR/NON-PD") and the assessment is less than ref_start_window days after reference_date.

- "ND": An assessment is considered as not done (ND) if AVALC == "ND".
- "MISSING": An assessment is considered as missing (MISSING) if a subject has no observation in the input dataset.

  If the missing_as_ne argument is set to TRUE, AVALC is set to "NE" for these subjects.

3. For each subject the best response as derived in the previous step is selected, where "CR" is best and "MISSING" is worst in the order above. If the best response is not unique, the first one (with respect to ADT) is selected. If the selected record is from the input dataset, all variables are kept. If the selected record is from dataset_adsl, all variables which are in both dataset and dataset_adsl are kept.

4. The AVAL variable is added and set to aval_fun(AVALC).

5. The variables specified by the set_values_to parameter are added to the new observations.

6. The new observations are added to input dataset.

Value

The input dataset with a new parameter for confirmed best overall response

Author(s)

Stefan Bundfuss

See Also

Other superseded: derive_param_bor(), derive_param_clinbenefit(), derive_param_confirmed_resp(), derive_param_response(), filter_pd()

Examples

```
library(dplyr)
library(lubridate)
library(admiral)

# Create ADSL dataset
adsl <- tibble::tribble(
  ~USUBJID, ~TRTSDTC,
  "1",      "2020-01-01",
  "2",      "2019-12-12",
  "3",      "2019-11-11",
  "4",      "2019-12-30",
  "5",      "2020-01-01",
  "6",      "2020-02-02",
  "7",      "2020-02-02",
  "8",      "2020-04-01",
  "9",      "2020-03-01"
) %>%
  mutate(
    TRTSDT = ymd(TRTSDTC),
    STUDYID = "XX1234"
  )
```

```
# Create ADRS dataset
ovr_obs <- tibble::tribble(
  ~USUBJID, ~ADTC,        ~AVALC,
  "1",      "2020-01-01", "PR",
  "1",      "2020-02-01", "CR",
  "1",      "2020-02-16", "NE",
  "1",      "2020-03-01", "CR",
  "1",      "2020-04-01", "SD",
  "2",      "2020-01-01", "SD",
  "2",      "2020-02-01", "PR",
  "2",      "2020-03-01", "SD",
  "2",      "2020-03-13", "CR",
  "3",      "2019-11-12", "CR",
  "3",      "2019-12-02", "CR",
  "3",      "2020-01-01", "SD",
  "4",      "2020-01-01", "PR",
  "4",      "2020-03-01", "SD",
  "4",      "2020-04-01", "SD",
  "4",      "2020-05-01", "PR",
  "4",      "2020-05-15", "NON-CR/NON-PD",
  "5",      "2020-01-01", "PR",
  "5",      "2020-01-10", "SD",
  "5",      "2020-01-20", "PR",
  "5",      "2020-05-15", "NON-CR/NON-PD",
  "6",      "2020-02-06", "PR",
  "6",      "2020-02-16", "CR",
  "6",      "2020-03-30", "PR",
  "6",      "2020-04-12", "PD",
  "6",      "2020-05-01", "CR",
  "6",      "2020-06-01", "CR",
  "7",      "2020-02-06", "PR",
  "7",      "2020-02-16", "CR",
  "7",      "2020-04-01", "NE",
  "9",      "2020-03-16", "CR",
  "9",      "2020-04-01", "NE",
  "9",      "2020-04-16", "NE",
  "9",      "2020-05-01", "CR"
) %>%
  mutate(PARAMCD = "OVR", ANL01FL = "Y")

pd_obs <-
  bind_rows(tibble::tribble(
    ~USUBJID, ~ADTC,        ~AVALC,
    "6",      "2020-04-12", "Y"
  ) %>%
    mutate(PARAMCD = "PD", ANL01FL = "Y"))

adrs <- bind_rows(ovr_obs, pd_obs) %>%
  mutate(
    ADT = ymd(ADTC),
    STUDYID = "XX1234"
  ) %>%
  select(-ADTC) %>%
```

```
  derive_vars_merged(
    dataset_add = adsl,
    by_vars = exprs(STUDYID, USUBJID),
    new_vars = exprs(TRTSDT)
  )

pd_date <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "PD" & ANL01FL == "Y"
)

# Derive confirmed best overall response parameter
derive_param_confirmed_bor(
  adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd_date,
  source_datasets = list(adrs = adrs),
  reference_date = TRTSDT,
  ref_start_window = 28,
  ref_confirm = 28,
  set_values_to = exprs(
    PARAMCD = "CBOR",
    PARAM = "Best Confirmed Overall Response by Investigator"
  )
) %>%
  filter(PARAMCD == "CBOR")

# Derive confirmed best overall response parameter (accepting SD for PR,
# accept two NEs, and considering missings as NE)
derive_param_confirmed_bor(
  adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd_date,
  source_datasets = list(adrs = adrs),
  reference_date = TRTSDT,
  ref_start_window = 28,
  ref_confirm = 28,
  max_nr_ne = 2,
  accept_sd = TRUE,
  missing_as_ne = TRUE,
  set_values_to = exprs(
    PARAMCD = "CBOR",
    PARAM = "Best Confirmed Overall Response by Investigator"
  )
) %>%
  filter(PARAMCD == "CBOR")
```

derive_param_confirmed_resp

Adds a Parameter for Confirmed Response

Description

[Superseded] The `derive_param_confirmed_resp()` function has been superseded in favor of `derive_extreme_event()`.

Adds a parameter for confirmed response

Usage

```
derive_param_confirmed_resp(
  dataset,
  dataset_adsl,
  filter_source,
  source_pd = NULL,
  source_datasets = NULL,
  ref_confirm,
  max_nr_ne = 1,
  accept_sd = FALSE,
  aval_fun,
  set_values_to,
  subject_keys = get_admiral_option("subject_keys")
)
```

Arguments

dataset	Input dataset
	The `PARAMCD`, `ADT`, and `AVALC` variables and the variables specified by `subject_keys` and `reference_date` are expected.
	After applying `filter_source` and/or `source_pd` the variable `ADT` and the variables specified by `subject_keys` must be a unique key of the dataset.
dataset_adsl	ADSL input dataset
	The variables specified for `subject_keys` are expected. For each subject of the specified dataset a new observation is added to the input dataset.
filter_source	Source filter
	All observations in `dataset_source` fulfilling the specified condition are considered for deriving the confirmed response.
source_pd	Date of first progressive disease (PD)
	If the parameter is specified, the observations of the input dataset for deriving the new parameter are restricted to observations up to the specified date. Observations at the specified date are included. For subjects without first PD date all observations are take into account.
	Permitted Values: a date_source object (see `admiral::date_source()` for details)

source_datasets

>Source dataset for the first PD date
>
>A named list of datasets is expected. It links the dataset_name from source_pd with an existing dataset.
>
>For example if source_pd = pd_date with
>
>```
>pd_date <- date_source(
>  dataset_name = "adrs",
>  date = ADT,
>  filter = PARAMCD == PD
>)
>```
>
>and the actual response dataset in the script is myadrs, source_datasets = list(adrs = myadrs) should be specified.

ref_confirm

>Minimum time period for confirmation
>
>The assessment and the confirmatory assessment for "CR" and "PR" have to be at least the specified number of days apart.

max_nr_ne

>The specified number of "NE" assessments between the assessment and the confirmatory assessment for "CR" and "PR" response is accepted.
>
>*Permitted Values:* a non-negative numeric scalar

accept_sd

>Accept "SD" for "PR"?
>
>If the argument is set to TRUE, one "SD" assessment between the assessment and the confirmatory assessment for "PR" response is accepted. Otherwise, no "SD" assessment must occur between the two assessments.
>
>*Permitted Values:* a logical scalar

aval_fun

>*Deprecated*, please use set_values_to instead.
>
>Function to map character analysis value (AVALC) to numeric analysis value (AVAL)
>
>The (first) argument of the function must expect a character vector and the function must return a numeric vector.

set_values_to

>Variables to set
>
>A named list returned by exprs() defining the variables to be set for the new parameter, e.g. exprs(PARAMCD = "CRSP", PARAM = "Confirmed Response") is expected. The values must be symbols, character strings, numeric values, or NA.

subject_keys

>Variables to uniquely identify a subject
>
>A list of symbols created using exprs() is expected.

Details

1. The input dataset (dataset) is restricted to the observations matching filter_source and to observations before or at the date specified by source_pd.

2. A subject is considered as responder if there is at least one observation in the restricted dataset with

   - AVALC == "CR",

- there is a confirmatory assessment with AVALC == "CR" at least ref_confirm days after the assessment,
- all assessments between the assessment and the confirmatory assessment are "CR" or "NE", and
- there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment.

or at least one observation with

- AVALC == "PR",
- there is a confirmatory assessment with AVALC %in% c("CR", "PR") at least ref_confirm days after the assessment,
- all assessments between the assessment and the confirmatory assessment are "CR", "PR", "SD", or "NE",
- there is no "PR" assessment after a "CR" assessment in the confirmation period,
- there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment,
- if the accept_sd argument is set to TRUE, one "SD" assessment in the confirmation period is accepted. Otherwise, no "SD" assessment must occur within the confirmation period.

3. For responders AVALC is set to "Y" and ADT to the first date where the response criteria are fulfilled. For all other subjects in dataset_adsl AVALC is set to "N" and ADT to NA.

4. The AVAL variable is added and set to aval_fun(AVALC).

5. The variables specified by the set_values_to parameter are added to the new observations.

6. The new observations are added to input dataset.

Value

The input dataset with a new parameter for confirmed response

Author(s)

Stefan Bundfuss

See Also

Other superseded: [derive_param_bor()](), [derive_param_clinbenefit()](), [derive_param_confirmed_bor()](), [derive_param_response()](), [filter_pd()]()

Examples

```
library(dplyr)
library(admiral)

# Create ADSL dataset
adsl <- tibble::tribble(
  ~USUBJID, ~TRTSDTC,
  "1",      "2020-01-01",
  "2",      "2019-12-12",
  "3",      "2019-11-11",
```

```
   "4",       "2019-12-30",
   "5",       "2020-01-01",
   "6",       "2020-02-02",
   "7",       "2020-02-02",
   "8",       "2020-04-01",
   "9",       "2020-03-01"
 ) %>%
   mutate(
     STUDYID = "XX1234"
   )

 # Create ADRS dataset
 ovr_obs <- tibble::tribble(
   ~USUBJID, ~ADTC,         ~AVALC,
   "1",      "2020-01-01", "PR",
   "1",      "2020-02-01", "CR",
   "1",      "2020-02-16", "NE",
   "1",      "2020-03-01", "CR",
   "1",      "2020-04-01", "SD",
   "2",      "2020-01-01", "SD",
   "2",      "2020-02-01", "PR",
   "2",      "2020-03-01", "SD",
   "2",      "2020-03-13", "CR",
   "3",      "2019-11-12", "CR",
   "3",      "2019-12-02", "CR",
   "3",      "2020-01-01", "SD",
   "4",      "2020-01-01", "PR",
   "4",      "2020-03-01", "SD",
   "4",      "2020-04-01", "SD",
   "4",      "2020-05-01", "PR",
   "4",      "2020-05-15", "NON-CR/NON-PD",
   "5",      "2020-01-01", "PR",
   "5",      "2020-01-10", "SD",
   "5",      "2020-01-20", "PR",
   "5",      "2020-05-15", "NON-CR/NON-PD",
   "6",      "2020-02-06", "PR",
   "6",      "2020-02-16", "CR",
   "6",      "2020-03-30", "PR",
   "6",      "2020-04-12", "PD",
   "6",      "2020-05-01", "CR",
   "6",      "2020-06-01", "CR",
   "7",      "2020-02-06", "PR",
   "7",      "2020-02-16", "CR",
   "7",      "2020-04-01", "NE",
   "9",      "2020-03-16", "CR",
   "9",      "2020-04-01", "NE",
   "9",      "2020-04-16", "NE",
   "9",      "2020-05-01", "CR"
 ) %>%
   mutate(PARAMCD = "OVR", ANL01FL = "Y")

 pd_obs <-
   bind_rows(tibble::tribble(
```

```
    ~USUBJID, ~ADTC,         ~AVALC,
    "6",       "2020-04-12", "Y"
  ) %>%
    mutate(PARAMCD = "PD", ANL01FL = "Y"))

adrs <- bind_rows(ovr_obs, pd_obs) %>%
  mutate(
    ADT = lubridate::ymd(ADTC),
    STUDYID = "XX1234"
  ) %>%
  select(-ADTC)

pd_date <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "PD" & ANL01FL == "Y"
)

# Derive confirmed response parameter
derive_param_confirmed_resp(
  adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd_date,
  source_datasets = list(adrs = adrs),
  ref_confirm = 28,
  set_values_to = exprs(
    PARAMCD = "CRSP",
    PARAM = "Confirmed Response by Investigator"
  )
) %>%
  filter(PARAMCD == "CRSP")

# Derive confirmed response parameter (accepting SD for PR and two NEs)
derive_param_confirmed_resp(
  adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd_date,
  source_datasets = list(adrs = adrs),
  ref_confirm = 28,
  max_nr_ne = 2,
  accept_sd = TRUE,
  set_values_to = exprs(
    PARAMCD = "CRSP",
    PARAM = "Confirmed Response by Investigator"
  )
) %>%
  filter(PARAMCD == "CRSP")
```

derive_param_response   *Adds a Parameter Indicating If a Subject Had a Response before Progressive Disease*

Description

[Superseded] The `derive_param_response()` function has been superseded in favor of `derive_extreme_event()`.

Adds a parameter indicating if a response has been observed. If a response has been observed, `AVALC` is set to "Y", `AVAL` to 1 and `ADT` is set to the first date when a response has been observed. If a response has not been observed, `AVALC` is set to "N", `AVAL` to 0 and `ADT` is set NA.

Usage

```
derive_param_response(
  dataset,
  dataset_adsl,
  filter_source,
  source_pd = NULL,
  source_datasets = NULL,
  set_values_to,
  aval_fun,
  subject_keys = get_admiral_option("subject_keys")
)
```

Arguments

`dataset`	Input dataset
	The variables specified by the `subject_keys`and `ADT` are expected.
	After applying `filter_source` and/or `source_pd` the variable `ADT` and the variables specified by `subject_keys` must be a unique key of the dataset.
`dataset_adsl`	Input dataset
	• The variables specified for `subject_keys` are expected.
	• For each observation of the specified dataset a new observation is added to the input dataset. This is to capture those patients that may never have had a tumor assessment.
`filter_source`	Source filter
	All observations in the `dataset` data fulfilling the specified condition are selected.
`source_pd`	Sources and conditions defining the end of the assessment period for the responses.
	An object of type `date_source` is expected
	All observations in `dataset` defining the response data fulfilling the `filter_source` condition are considered as response if they fall before the end of the assessment period as defined by `source_pd`.

- For subjects with at least one response before the end of the assessment period, AVALC is set to "Y", AVAL to 1, and ADT to the first date when the response occurred.
- For all other subjects AVALC is set to "N", AVAL to 0, and ADT to NA.

source_datasets

Source dataset

A named list of datasets with one element is expected (e.g. list(adrs= adrs)).

The name must match the dataset_name field of the admiral::date_source() object specified for source_pd.

The variables specified by the subject_keys argument and the date field of the admiral::date_source() object are expected in the dataset.

set_values_to   Variables to set

A named list returned by exprs() defining the variables to be set for the new parameter, e.g. exprs(PARAMCD = "RSP", PARAM = "Response by investigator") is expected.

The values must be symbols, character strings, numeric values or NA.

aval_fun   *Deprecated*, please use set_values_to instead.

Function to map character analysis value (AVALC) to numeric analysis value (AVAL)

The (first) argument of the function must expect a character vector and the function must return a numeric vector.

subject_keys   Variables to uniquely identify a subject

A list of symbols created using exprs() is expected.

Details

1. The Date of the end of the assessment period (e.g. Progressive disease, as defined by pd_source) is added to the response dataset.

2. The response dataset is restricted to observations occurring before **or on** the date of progressive disease.

3. For each subject (with respect to the variables specified for the subject_keys parameter), the first observation (with respect to ADT) where the response condition (filter_source parameter) is fulfilled is selected.

4. For each observation in dataset_adsl a new observation is created.

   - For subjects with a response AVALC is set to "Y", AVAL to 1, and ADT to the first date (ADT) where the response condition is fulfilled.
   - For all other subjects AVALC is set to "N", AVAL to 0 and ADT to NA.

5. The variables specified by the set_values_to parameter are added to the new observations.

6. The new observations are added to input dataset.

Value

The input dataset with a new parameter indicating if and when a response occurred

Author(s)

Samia Kabi

See Also

Other superseded: derive_param_bor(), derive_param_clinbenefit(), derive_param_confirmed_bor(),
derive_param_confirmed_resp(), filter_pd()

Examples

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

adsl <- tribble(
  ~USUBJID,
  "1",
  "2",
  "3",
  "4"
) %>%
  mutate(STUDYID = "XX1234")

adrs <- tribble(
  ~USUBJID, ~PARAMCD, ~ADTC,          ~AVALC, ~ANL01FL,
  "1",      "OVR",    "2020-01-02",   "PR",   "Y",
  "1",      "OVR",    "2020-02-01",   "CR",   "Y",
  "1",      "OVR",    "2020-03-01",   "CR",   "Y",
  "1",      "OVR",    "2020-04-01",   "SD",   "Y",
  "1",      "PD",     NA_character_,  "N",    "Y",
  "2",      "OVR",    "2021-06-15",   "SD",   "Y",
  "2",      "OVR",    "2021-07-16",   "PD",   "Y",
  "2",      "OVR",    "2021-09-14",   "PD",   "Y",
  "2",      "PD",     "2021-09-14",   "Y",    "Y",
  "3",      "OVR",    "2021-09-14",   "SD",   "Y",
  "3",      "OVR",    "2021-10-30",   "PD",   "Y",
  "3",      "OVR",    "2021-12-25",   "CR",   "Y",
  "3",      "PD",     "2021-10-30",   "Y",    "Y"
) %>%
  mutate(
    STUDYID = "XX1234",
    ADT = ymd(ADTC),
    ANL01FL = "Y"
  ) %>%
  select(-ADTC)

# Define the end of the assessment period for responses:
# all responses before or on the first PD will be used.
pd <- date_source(
  dataset_name = "adrs",
  date = ADT,
```

```
    filter = PARAMCD == "PD" & AVALC == "Y"
)
# Derive the response parameter
derive_param_response(
  dataset = adrs,
  dataset_adsl = adsl,
  filter_source = PARAMCD == "OVR" & AVALC %in% c("CR", "PR") & ANL01FL == "Y",
  source_pd = pd,
  source_datasets = list(adrs = adrs),
  set_values_to = exprs(
    AVAL = yn_to_numeric(AVALC),
    PARAMCD = "RSP",
    PARAM = "Response by investigator"
  ),
  subject_keys = get_admiral_option("subject_keys")
) %>%
  arrange(USUBJID, PARAMCD, ADT)
```

---

filter_pd   *Filter up to First PD (Progressive Disease) Date*

Description

[Superseded] The `filter_pd()` function has been superseded in favor of `filter_relative()`.

Filter a dataset to only include the source parameter records up to and including the first PD (progressive disease). These records are passed to downstream derivations regarding responses such as BOR (best overall response).

Usage

```
filter_pd(
  dataset,
  filter,
  source_pd,
  source_datasets,
  subject_keys = get_admiral_option("subject_keys")
)
```

Arguments

dataset	Input dataset
	The variables ADT and those specified by `subject_keys` are expected.
filter	Filter condition for restricting the input dataset
source_pd	A `admiral::date_source()` object providing the date of first PD
	For each subject the first date (`date` field) in the provided dataset (`dataset_name` field) restricted by `filter` field is considered as first PD date.

source_datasets

> A named list of data sets is expected.
>
> The name must match the name provided by the dataset_name field of the admiral::date_source() object specified for source_pd.

subject_keys   Variables to uniquely identify a subject

> A list of symbols created using exprs() is expected.

Details

1. The input dataset (dataset) is restricted by filter.

2. For each subject the first PD date is derived as the first date (source_pd$date) in the source pd dataset (source_datasets[[source_pd$dataset_name]]) restricted by source_pd$filter.

3. The restricted input dataset is restricted to records up to first PD date. Records matching first PD date are included. For subject without any first PD date, all records are included.

Value

A subset of the input dataset

Author(s)

Teckla Akinyi, Stefan Bundfuss

See Also

Other superseded: derive_param_bor(), derive_param_clinbenefit(), derive_param_confirmed_bor(), derive_param_confirmed_resp(), derive_param_response()

Examples

```
library(dplyr)
library(lubridate)
library(admiral)
library(admiralonco)

# Filter OVR records up to first PD, first PD date provided in separate BDS dataset (adevent)
adrs <- tibble::tribble(
  ~STUDYID,        ~USUBJID,       ~PARAMCD, ~AVALC, ~ADT,          ~ANL01FL,
  "CDISCPILOT01", "01-701-1015", "OVR",    "CR",   "2016-01-25", "Y",
  "CDISCPILOT01", "01-701-1015", "OVR",    "SD",   "2016-02-22", NA_character_,
  "CDISCPILOT01", "01-701-1015", "OVR",    "PD",   "2016-02-22", "Y",
  "CDISCPILOT01", "01-701-1015", "BOR",    "CR",   "2016-01-25", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",    "SD",   "2015-12-07", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",    "PD",   "2016-04-25", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",    "PD",   "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1034", "BOR",    "SD",   "2015-12-07", "Y",
  "CDISCPILOT01", "01-701-1035", "OVR",    "SD",   "2016-04-25", "Y",
  "CDISCPILOT01", "01-701-1035", "OVR",    "PR",   "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1035", "BOR",    "PR",   "2016-06-25", "Y"
) %>% mutate(
```

```
  ADT = as_date(ADT)
)

adevent <- tibble::tribble(
  ~STUDYID,        ~USUBJID,       ~PARAMCD, ~AVALC, ~ADT,
  "CDISCPILOT01", "01-701-1015", "PD",      "Y",    "2016-02-22",
  "CDISCPILOT01", "01-701-1034", "PD",      "Y",    "2016-04-25"
) %>% mutate(
  ADT = as_date(ADT)
)

pd <- date_source(
  dataset_name = "adevent",
  date = ADT,
  filter = PARAMCD == "PD"
)

filter_pd(
  dataset = adrs,
  filter = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd,
  source_datasets = list(adevent = adevent)
)

# Filter OVR records up to first PD, first PD date provided in ADSL dataset
adsl <- tibble::tribble(
  ~STUDYID,        ~USUBJID,       ~PDDT,
  "CDISCPILOT01", "01-701-1015", "2016-02-22",
  "CDISCPILOT01", "01-701-1034", "2016-04-25",
  "CDISCPILOT01", "01-701-1035", ""
) %>% mutate(
  PDDT = as_date(PDDT)
)

pd <- date_source(
  dataset_name = "adsl",
  date = PDDT
)

filter_pd(
  dataset = adrs,
  filter = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd,
  source_datasets = list(adsl = adsl)
)

# Filter OVR records up to first PD, first PD date provided in input dataset (PD parameter)
adrs <- tibble::tribble(
  ~STUDYID,        ~USUBJID,       ~PARAMCD, ~AVALC, ~ADT,          ~ANL01FL,
  "CDISCPILOT01", "01-701-1015", "OVR",     "CR",   "2016-01-25", "Y",
  "CDISCPILOT01", "01-701-1015", "OVR",     "SD",   "2016-02-22", NA_character_,
  "CDISCPILOT01", "01-701-1015", "OVR",     "PD",   "2016-02-22", "Y",
  "CDISCPILOT01", "01-701-1015", "BOR",     "CR",   "2016-01-25", "Y",
```

```
  "CDISCPILOT01", "01-701-1034", "OVR",     "SD",    "2015-12-07", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",     "PD",    "2016-04-25", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",     "PD",    "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1034", "BOR",     "SD",    "2015-12-07", "Y",
  "CDISCPILOT01", "01-701-1035", "OVR",     "SD",    "2016-04-25", "Y",
  "CDISCPILOT01", "01-701-1035", "OVR",     "PR",    "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1035", "BOR",     "PR",    "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1015", "PD",      "Y",     "2016-02-22", "Y",
  "CDISCPILOT01", "01-701-1034", "PD",      "Y",     "2016-04-25", "Y"
) %>% mutate(
  ADT = as_date(ADT)
)

pd <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "PD"
)

filter_pd(
  dataset = adrs,
  filter = PARAMCD == "OVR" & ANL01FL == "Y",
  source_pd = pd,
  source_datasets = list(adrs = adrs)
)

# Filter OVR records up to first PD, first PD date derived from OVR records
adrs <- tibble::tribble(
  ~STUDYID,        ~USUBJID,      ~PARAMCD, ~AVALC, ~ADT,          ~ANL01FL,
  "CDISCPILOT01", "01-701-1015", "OVR",     "CR",    "2016-01-25", "Y",
  "CDISCPILOT01", "01-701-1015", "OVR",     "SD",    "2016-02-22", NA_character_,
  "CDISCPILOT01", "01-701-1015", "OVR",     "PD",    "2016-02-22", "Y",
  "CDISCPILOT01", "01-701-1015", "BOR",     "CR",    "2016-01-25", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",     "SD",    "2015-12-07", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",     "PD",    "2016-04-25", "Y",
  "CDISCPILOT01", "01-701-1034", "OVR",     "PD",    "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1034", "BOR",     "SD",    "2015-12-07", "Y",
  "CDISCPILOT01", "01-701-1035", "OVR",     "SD",    "2016-04-25", "Y",
  "CDISCPILOT01", "01-701-1035", "OVR",     "PR",    "2016-06-25", "Y",
  "CDISCPILOT01", "01-701-1035", "BOR",     "PR",    "2016-06-25", "Y"
) %>% mutate(
  ADT = as_date(ADT)
)

pd <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "OVR" & ANL01FL == "Y" & AVALC == "PD"
)

filter_pd(
  dataset = adrs,
  filter = PARAMCD == "OVR" & ANL01FL == "Y",
```

```
    source_pd = pd,
    source_datasets = list(adrs = adrs)
)
```

---

get_crpr_dataset   *Get CR Records Followed by PR That Lead to a Prior Error*

Description

Get CR Records Followed by PR That Lead to a Prior Error

Usage

```
get_crpr_dataset()
```

Details

Some {admiralonco} functions check that in the source records CR is not followed by PR and throw an error otherwise. The get_crpr_dataset() function allows one to retrieve the duplicate records that lead to an error.

Note that the function always returns the dataset of duplicates from the last error that has been thrown in the current R session. Thus, after restarting the R sessions get_crpr_dataset() will return NULL and after a second error has been thrown, the dataset of the first error can no longer be accessed (unless it has been saved in a variable).

Value

A data.frame or NULL

Author(s)

Stefan Bundfuss

See Also

signal_crpr()

Utilities for Dataset Checking: signal_crpr()

Examples

```
library(tibble)
library(dplyr)
library(lubridate)
library(admiralonco)
library(rlang)

adrs <- tribble(
  ~USUBJID, ~ADTC,          ~AVALC,
```

```
    "1",        "2020-01-01", "PR",
    "1",        "2020-02-01", "CR",
    "1",        "2020-02-16", "NE",
    "1",        "2020-03-01", "CR",
    "2",        "2020-02-06", "PR",
    "2",        "2020-02-16", "CR",
    "2",        "2020-03-30", "PR",
) %>%
  mutate(
    ADT = ymd(ADTC),
    STUDYID = "XX1234"
  )

signal_crpr(adrs, order = exprs(ADT))

get_crpr_dataset()
```

---

rsp_y   *Pre-Defined Response Event Objects*

Description

These pre-defined event() and event_joined() objects can be used as input to admiral::derive_extreme_event().

Usage

```
rsp_y

no_data_n

cb_y

bor_cr

bor_pr

bor_sd

bor_non_crpd

bor_pd

bor_ne

no_data_missing

crsp_y_cr
```

```
crsp_y_pr

cbor_cr

cbor_pr
```

Details

To see the definition of the various objects simply print the object in the R console, e.g. `bor_sd`. For details of how to use these objects please refer to [admiral::derive_extreme_event()](#).

It is assumed that `dataset_name = "ovr"` refers to the dataset of the only overall response assessments at each visit which should be considered for the parameter derivations. For example the dataset should include only post-baseline assessments up to first PD and before start of anti-cancer therapy.

See Also

[admiral::derive_extreme_event()](#), [admiral::event()](#), [admiral::event_joined()](#)

Examples

```
# This shows the definition of all pre-defined `event` objects that ship
# with {admiralonco}
exports <- sort(getNamespaceExports("admiralonco"))
for (obj_name in exports) {
  obj <- getExportedValue("admiralonco", obj_name)
  if (inherits(obj, "event_def")) {
    cat("\n", obj_name, ":\n", sep = "")
    print(obj, indent = 2)
  }
}
```

---

signal_crpr   *Signal CR Records Followed by PR*

Description

Signal CR Records Followed by PR

Usage

```
signal_crpr(
  dataset,
  order,
  msg = "Dataset contains CR records followed by PR.",
  subject_keys = get_admiral_option("subject_keys"),
  check_type = "warning"
)
```

Arguments

dataset	A data frame
order	A list of variables created using exprs() determining the order or the records
msg	The condition message
subject_keys	Variables to uniquely identify a subject
	A list of symbols created using exprs() is expected.
check_type	Type of message to issue when detecting PR after CR.
	Permitted Values: "message", "warning" or "error"

Value

No return value, called for side effects

Author(s)

Stefan Bundfuss

See Also

[get_crpr_dataset()](#)

Utilities for Dataset Checking: [get_crpr_dataset](#)()

Examples

```
library(tibble)
library(dplyr)
library(lubridate)
library(admiralonco)
library(rlang)

adrs <- tribble(
  ~USUBJID, ~ADTC,        ~AVALC,
  "1",      "2020-01-01", "PR",
  "1",      "2020-02-01", "CR",
  "1",      "2020-02-16", "NE",
  "1",      "2020-03-01", "CR",
  "2",      "2020-02-06", "PR",
  "2",      "2020-02-16", "CR",
  "2",      "2020-03-30", "PR",
) %>%
  mutate(
    ADT = ymd(ADTC),
    STUDYID = "XX1234"
  )

signal_crpr(adrs, order = exprs(ADT))
```

Index