Supplementary Material

A Process for the Emulation of Comparative Oncology Trials with Real-world Evidence (ENCORE)

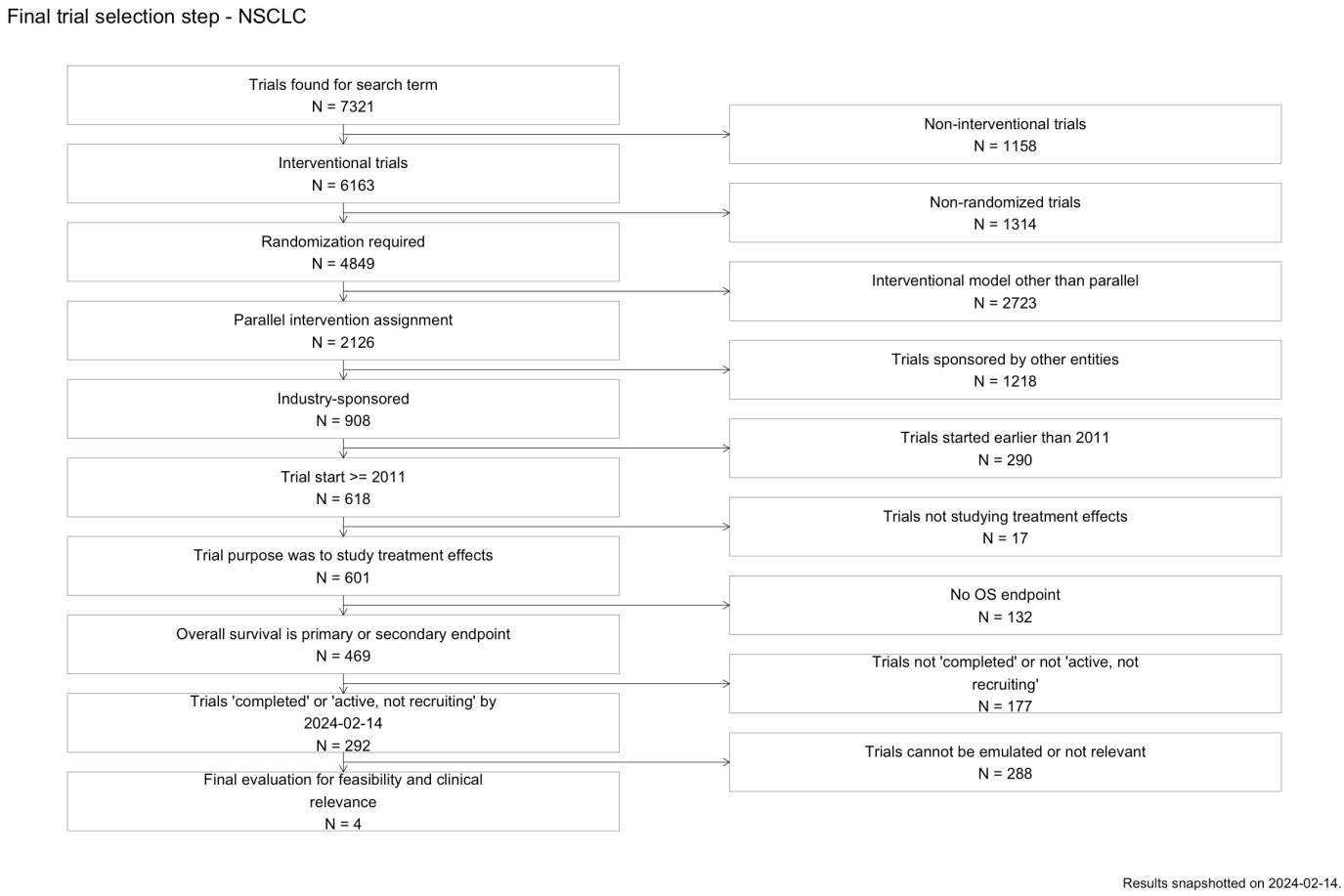
Table of contents

Supplementary Figures **2**

R package documentation **6**

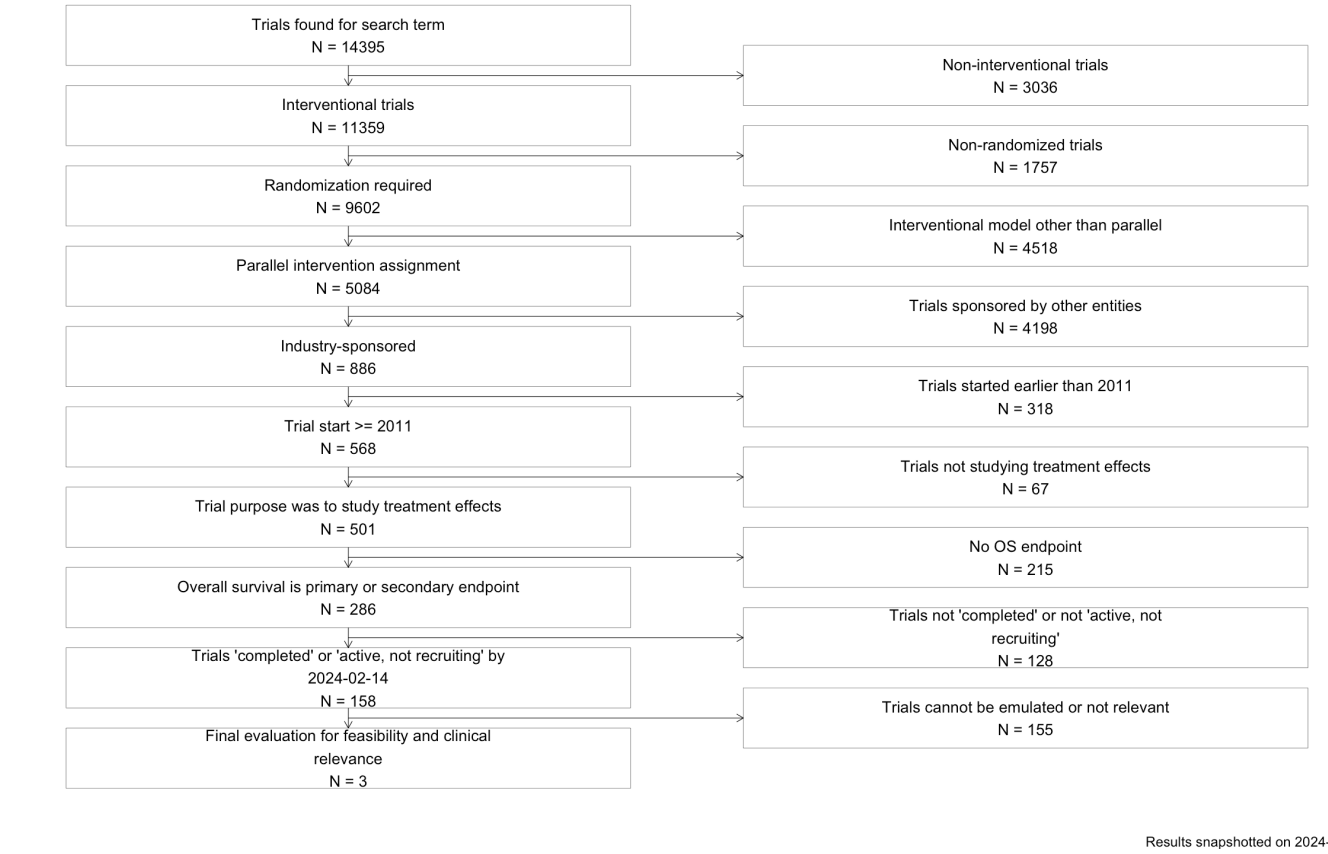
Supplementary Figures

Supplementary Figure 1: CONSORT diagram for non-small cell lung cancer (NSCLC) top candidate trials.

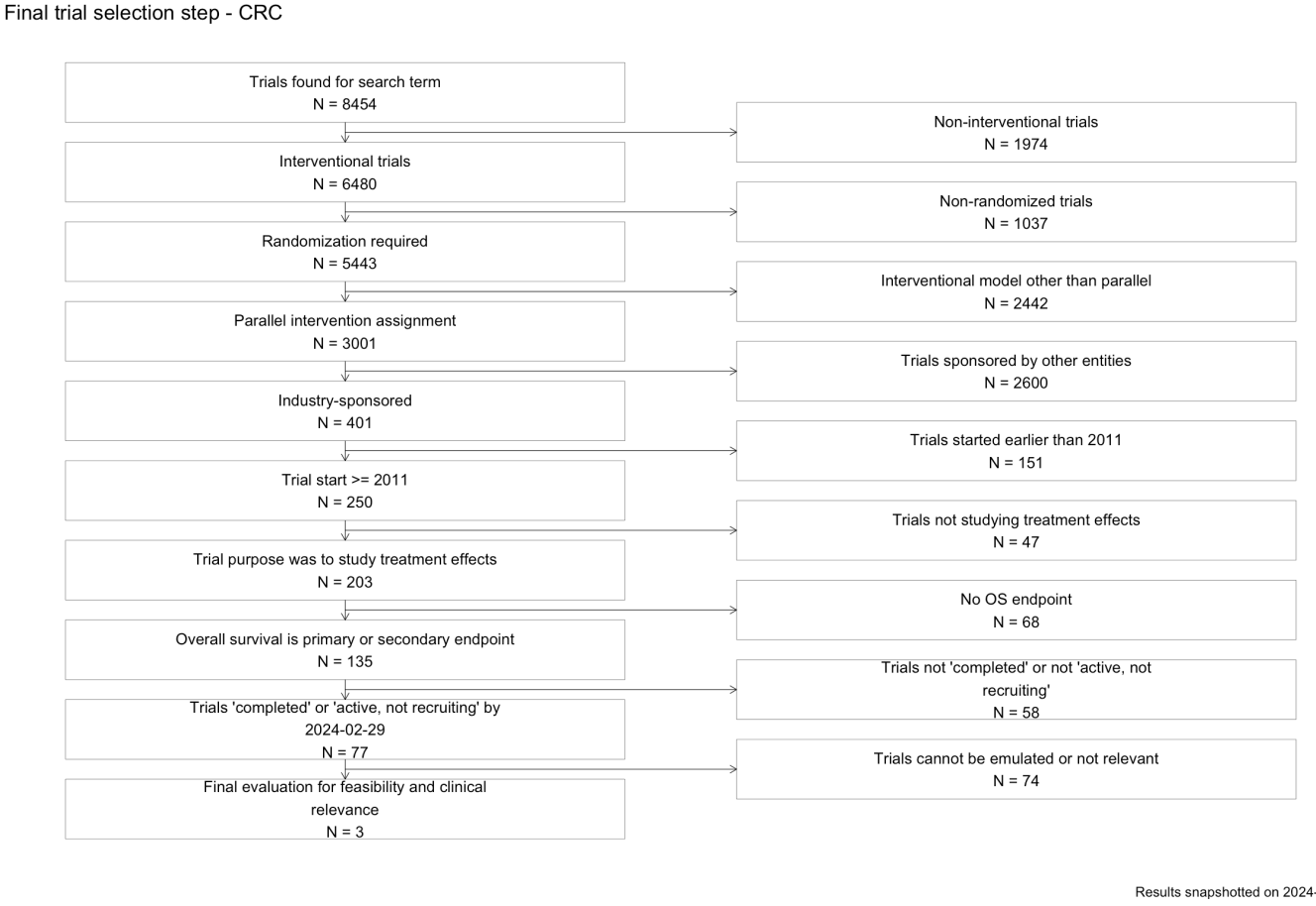


Note that Supplementary Figure 1 results in four shortlisted candidates because both CheckMate017/057 have been both shortlisted and only differ in the squamous versus nonsquamous histological eligibility of the trial population.

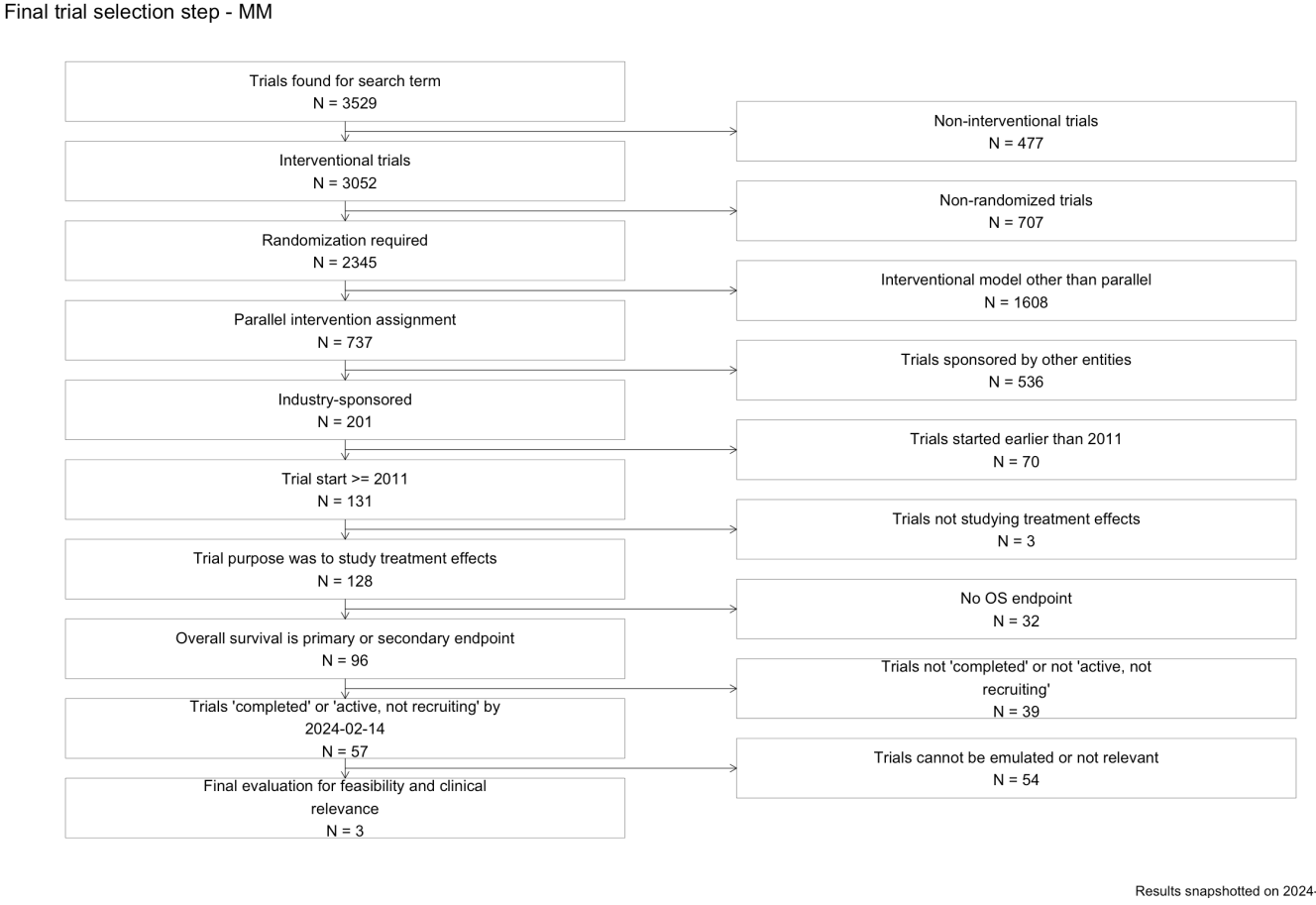
Supplementary Figure 2: CONSORT diagram for breast cancer (BC) top candidate trials.



Supplementary Figure 3: CONSORT diagram for colorectal cancer (CRC) top candidate trials.



Supplementary Figure 4: CONSORT diagram for multiple myeloma (MM) top candidate trials.



R package documentation

The following documentation for the internal `encore.io` packages describes and details reproducible functions to query analytic cohorts consistently across trial emulations.

> **!** Important
>
> The `encore.io` package will be continually developed throughout the ENCORE project and the following documentation is a version-controlled snapshot at the time of this publication. Updated versions will be published with the corresponding protocols for each trial emulation separately.

Package 'encore.io'

March 5, 2025

Type Package

Title Functions and Wrappers To Streamline Analytics For The ENCORE Trial Emulation Project

Version 0.2.0

Description This package contains important functions to streamline the analytics for the ENCORE trial emulation project. This includes the query of eligible trials to emulate and most analytical steps to emulate these trials.

BugReports https://gitlab.partners.org/drugepi/encore/encore.io

License Apache License (>= 2)

Encoding UTF-8

LazyData true

Imports arrow,
assertthat,
data.table,
dplyr,
plyr,
forcats,
glue,
ggplot2,
gt,
gtsummary,
parallel,
stringr,
survival,
tibble,
tidyr,
lifecycle,
lubridate,
magrittr,
MatchIt,
mice,
pROC,
WeightIt

RoxygenNote 7.3.1

Suggests anesrake,
cobalt,
devtools,
DT,

here,
knitr,
locfit,
MatchThem,
rmarkdown,
scales,
simsurv,
smdi,
testthat (>= 3.0.0)

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 2.10)

Contents

create_table1 4
c_statistics 5
edb1_3_4_compute_ropro 6
edb1_cohorts 7
edb1_get_biomarker 7
edb1_get_demographics 10
edb1_get_diagnosis_heme 12
edb1_get_diagnosis_solid 14
edb1_get_ecog 16
edb1_get_histology 18
edb1_get_labs 19
edb1_get_os 22
edb1_get_vitals 24
edb1_query_ropro 26
edb2_assign_date 28
edb2_compute_ropro 29
edb2_get_biomarker 30
edb2_get_demographics 32
edb2_get_diagnosis_heme 34
edb2_get_diagnosis_solid 35
edb2_get_ecog 38
edb2_get_histology 40
edb2_get_labs 41
edb2_get_os 44
edb2_get_vitals 46
edb2_path_helper 48
edb2_query_ropro 49
edb3_get_biomarker 50
edb3_get_demographics 52
edb3_get_diagnosis_solid 53
edb3_get_ecog 55
edb3_get_histology 56
edb3_get_labs 57
edb3_get_os 60

edb3_get_vitals . . . 61
edb3_query_ropro . . . 63
edb4_get_biomarker . . . 65
edb4_get_demographics . . . 67
edb4_get_diagnosis_heme . . . 69
edb4_get_diagnosis_solid . . . 70
edb4_get_ecog . . . 72
edb4_get_histology . . . 74
edb4_get_labs . . . 76
edb4_get_os . . . 79
edb4_get_vitals . . . 80
edb4_query_ropro . . . 83
ess . . . 84
gt_tbl_compact . . . 85
icd_metastases . . . 86
imputation_workflow . . . 86
km_pooling . . . 87
labs_mapping_edb1 . . . 90
labs_mapping_edb2 . . . 91
labs_mapping_edb3 . . . 91
labs_mapping_edb4 . . . 92
labs_mapping_implausible_values . . . 92
n_fmt . . . 93
power_survival . . . 94
ps_balance_plot . . . 95
qc_assertive_line_check . . . 96
re_weight . . . 97
ropro_aNSCLC_covars . . . 99
ropro_covars_log_log_transform . . . 99
ropro_covars_log_transform . . . 100
ropro_earlyBreast_covars . . . 100
ropro_mBreast_covars . . . 100
ropro_mCRC_covars . . . 101
ropro_MM_covars . . . 101
ropro_pan_tumor_covars . . . 101
ropro_pan_tumor_covars_categorical . . . 102
simulate_flaura . . . 102
state_region_mapping . . . 103
vitals_mapping_edb1 . . . 103
vitals_mapping_edb2 . . . 104
vitals_mapping_edb3 . . . 104
vitals_mapping_edb4 . . . 105
vitals_mapping_implausible_values . . . 106

Index **107**

`create_table1` *Wrapper around* `gtsummary::tbl_summary()` *to create a beautiful Table 1's quickly*

Description

Create a table 1 fast.

Usage

```
create_table1(
  x = NULL,
  covariates = NULL,
  covariates_labels = NULL,
  treat = "treat",
  explicit_na_categorical = TRUE
)
```

Arguments

`x`	dataframe queried from edbx with treatment stratification variable and covariates to be displayed in the Table 1
`covariates`	character vector of columns/covariate names to be displayed in Table 1
`covariates_labels`	named character vector or list of formulas specifying variables labels of covariate-label pairs to display in table
`treat`	character specifying column name of treatment variable
`explicit_na_categorical`	logical, should missings in categorical variables be explicitly included as a separate category (default is TRUE)

Details

Wrapper for `tbl_summary`.

Value

object of class "tbl_summary" "gtsummary"

Examples

```
## Not run:
library(encore.io)
set global option to make gtsummary tables more compact
theme_gtsummary_compact()
table1 <- ard |>
 create_table1(
  covariate = table1_covariates$covariate,
  treat = "treat")


## End(Not run)
```

`c_statistics` *Calculates c-statistics for mimids/wimids objects*

Description

[Experimental] Calculates the propensity score c-statistics (= area under the curve) for imputed and unmatched/all and matched datasets resulting from `MatchThem::matchthem` (`mimids` `objects`)

Usage

```
c_statistics(
  object = NULL,
  exposure = "treat",
  weights = "weights",
  ps = "distance"
)
```

Arguments

`object`	mimids or data.frame object from complete(..., action = 'long', all = TRUE, ...)
`exposure`	character, quoted name of the exposure/treatment variable (must be of class `factor`)
`weights`	character, quoted name of the variable indicating the matching weights (usually 0: unmatched and 1: matched)
`ps`	character, quoted name of the variable with the distance measure (e.g., propensity score)

Details

The object input needs to be a mimids object or a data.frame object coming from MatchThem::matchthem(). If the mimids object is already converted to a long data.frame of stacked imputed datasets, the MatchThem::complete() function needs to be completed using action = "long" and all = TRUE arguments.

The function computes the c-statistic by computing the AUC in each imputed dataset and then summarizing the avaerge and min/max c-statistic by matching group. This aims to describe how well treatment can be predicted given a patient's propensity score. The idea is that in well-matched or weighted datasets, the c-statistic should be close to 0.5, i.e., we can't infer treatment propensity anymore given a patient's baseline covariates.

Value

tibble with summary c-statistics across imputed datasets

See Also

Franklin JM, Rassen JA, Ackermann D, Bartels DB, Schneeweiss S. Metrics for covariate balance in cohort studies of causal effects. Stat Med. 2014 May 10;33(10):1685-99. doi: 10.1002/sim.6058. Epub 2013 Dec 9. PMID: 24323618.

Examples

```
## Not run:
library(encore.io)

 c_statistics(
  object = edb1_mimids,
  exposure = "treat",
  ps = "distance"
  )


## End(Not run)
```

`edb1_3_4_compute_ropro` *Derive ROPRO prognostic score*

Description

This function computes ROPRO prognostic score (Becker, Weberpals, et al., Ann Oncol 2020) for a given inception cohort.

Usage

```
edb1_3_4_compute_ropro(
  x = NULL,
  cancer = c("aNSCLC", "MetastaticBreast", "EarlyBreast", "MetastaticCRC",
    "MultipleMyeloma")
)
```

Arguments

`x`	dataframe with inception cohort and required ROPRO covariates
`cancer`	character, what cancer-specific ROPRO should be computed ("aNSCLC", "MetastaticBreast", "EarlyBreast", "MetastaticCRC", "MultipleMyeloma")

Details

This function takes in a dataframe with all required variables to compute the general and cancer-specific ROPRO (specified in `cancer`). The variables need to be queried and transformed before which can be done through th `edb(x)_query_ropro` functions (specific for each database).

Important: This function is only valid for data coming from EDB1, EDB3 and EDB4. Since EDB2 does not have all required variables, please use the `edb2_compute_ropro` function for this database to compute a reduced ROPRO model.

Value

The function returns x with the final general and cancer-specific ROPRO

Examples

```
## Not run:
library(encore.io)

x_ropro <- x |>
  edb_4_query_ropro(
    index_date = "dt_index",
    path = Sys.getenv("path_edb4"),
    cancer = "NSCLC",
    max_lookback = -90,
    verbose = TRUE
    ) |>
 edb_1_3_4_compute_ropro(
    cancer = "aNSCLC"
  )


## End(Not run)
```

`edb1_cohorts` *System data used to streamline functions and analysis*

Description

System data used to streamline functions and analysis

Usage

```
edb1_cohorts
```

Format

`edb1_cohorts`**:**
A tibble with edb1-specific mappings

cancer Abbreviated cancer types

cohort Cancer type sub-directory

biomarker Cancer type biomarker tables

special_biomarker_cols Special biomarker columns to select ...

`edb1_get_biomarker` *Query biomarker information for a given inception cohort*

Description

Function queries biomarker tables and curates information on alterations in defined driver genes.

Usage

```
edb1_get_biomarker(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  biomarker_name = NULL,
  from = -90,
  to = 0,
  label_name = FALSE
)
```

Arguments

x	dataframe queried with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date
path	character string, path to directory where EDB1 data/files are located
cancer	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", or "MetastaticBreast"
biomarker_name	character, name of biomarker/gene alteration (see details)
from	integer, left boundary of biomarker measurement window relative to index date (e.g., -90, indicating biomarker should be measured not before 90 days before index date)
to	integer, right boundary of biomarker measurement window relative to index date (e.g., 0, indicating biomarker should be measured until day of index date (inclusive))
label_name	logical, should variable name carry information about the measurement window

Details

The function queries and categorizes a certain biomarker that was collected and curated as part of the database as either positive or negative. The categorization happens according to if the biomarker mutation/alteration is a clinically actionable one. For example, patients with any mutation in the EGFR gene or MSI-H/dMMR status would be classified as positive. In case there are multiple measurements per biomarker and patient, the time point of measurement is defined as the non-missing biomarker result in the covariate ascertainment window given by `from` and `to` that is closest to the `index_date`. The biomarker date (`dt_(biomarker)`) in this database is defined as the earliest of specimen collected date, specimen received date or result date.

Depending on the cancer type, the biomarker names can be: ALK, BRAF, EGFR, HER2/ERBB2, KRAS, MET, NTRK1, NTRK2, NTRK3, PDL1, RET, ROS1, NTRK - unknown gene type, NTRK - other, MMR/MSI, NRAS, ER, HER2, Ki-67, PR, BRCA, ESR1, Oncotype, Mammaprint, PIK3CA

The function operates via the following logic:

1. Pull biomarker table and subset to patients included in inception cohort `x`.
2. Subset table to biomarker names as specified in `biomarker_name`.
3. Biomarker status test result is categorized hierarchically:
    - If the lowercase biomarker status matches any of "equivocal" OR "pending" OR "unknown", the result is mapped to NA (missing).

- If the lowercase biomarker status matches any of "not present" OR "negative" OR "no brca mutation" OR "not amplified" OR "ihc 0" OR "msi-l" OR "mss" OR "low risk" OR "normal mmr", the result is mapped to "negative".
- If the lowercase biomarker status matches any of "present" OR "positive" OR "mutation" OR "polymorphism" OR "amplified" OR "mmr protein deficiency" OR "msi-h" OR "high risk", the result is mapped to "positive".

4. Missing mapped biomarker status results are discarded.
5. The biomarker date is specified as the earliest of the specimen collected date, specimen received date or result date.
6. Biomarker dates outside of the specified measurement window relative to the index date (defined using the `index_date`, `from` and `to` arguments) are discarded.
7. If multiple biomarker tests are available for a given patient, the prioritization step is carried out as follows:
   - For each patient, biomarker mappings (negative, positive) and the absolute distance from biomarker date to index date are sorted in descending and ascending order, respectively.
   - To prioritize any positive biomarker mapping closest to the index date, the first row for each patient is selected. This reflects the final mapped biomarker status variable.
   - In addition, all available biomarker details within the measurement window are collapsed into a new variable. This reflects the final biomarker detail variable.
8. The newly created biomarker variables are named accordingly and are joined back to `x`.

WARNING: the categorization of more complex biomarkers like HER2 requires a more refined approach or de novo written code to account for equivocal results.

Value

x with all additional biomarker variables joined, that is:

- `c_(biomarker_name)_status_(from)_(to)` (binary, biomarker mutation status positive or negative)
- `c_(biomarker_name)_detail_(from)_(to)` (character string, more details about selected measurement)
- `c_(biomarker_name)_detail_all_(from)_(to)` (character string, this provides details about all results and details for this biomarker if there were multiple tests in the measurement window (from, to))
- `c_(biomarker_name)_distance_(from)_(to)` (numeric, relative distance between date of measurement and index date in days)
- `dt_(biomarker_name)_(from)_(to)` (date, date of selected biomarker measurement)

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb1_get_biomarker(
    cancer = "aNSCLC",
    biomarker_name = "EGFR",
```

```
    from = -180,
    to = 0)


## End(Not run)
```

`edb1_get_demographics` *Query demographic variables for an inception cohort*

Description

Function queries all available demographic variables for a given inception cohort.

Usage

```
edb1_get_demographics(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL
)
```

Arguments

`x`	dataframe queried with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date
`path`	character string, path to directory where EDB1 data/files are located
`cancer`	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"

Details

The function queries and categorizes demographic variables as reported in the electronic health record (EHR). The function operates via the following logic:

Demographics:

1. Pull the initial diagnosis date of the primary cancer and subset to patients included in inception cohort `x`.
2. Pull the demographics table and join the diagnosis date and index date for each patient.
3. The patient date of birth is reported on year-level granularity and is imputed to the 2nd July of the respective year (e.g. 1955 becomes 1955-07-02).
4. The age at initial diagnosis and at index date is computed as
   - The date of initial diagnosis - imputed date of birth
   - The index date - imputed date of birth
5. Additional categorical age variables are computed as <65 and 65+ years of age.
6. The reported sex, race and ethnicity variables are provided in a one-row-per-patient format and directly mapped one-to-one.

7. Additionally, race and ethnicity are combined into a new combined variable `dem_race_ethnicity` with five mutually exclusive groups as defined by SEER. As opposed to other datasets, EDB1 has a larger "Other" race category since "American Indian or Alaska Native" seems to be not coded as such, i.e, "Other" is an explicit group in this case.
8. The patient's state of residence is mapped to one of the major US geographic regions(Northeast, South, Midwest, West) and the state-level mapping can be looked up by calling the `state_region_mapping` mapping table.
9. If a patient is treated in different practice types (i.e., both academic and community), the patient gets assigned a mixed practice type (academic, community); this is seen only rarely.
10. The newly created variables are joined back to `x`.

Smoking: For aNSCLC and EarlyNSCLC, patients are documented as having a smoking history, no smoking history or unknown/not documented smoking history with one unique EHR-recorded smoking status per patient. Patients are mapped accordingly to being a current or former smoker (= smoking history), never smoker (= no smoking history) or having a missing value (unknown/not documented smoking history).

Socio-economic status (SES): EDB1 records a unique socio-economic determinant of health variable for each patient. This variable provides the relevant SES Index quintile for a patient's block group based on 2015-2019 Census data (1-lowest SES, 2, 3, 4, 5- highest SES).

Value

x with all additional demographic variables joined (see details), that is,

- dem_age_initial_diagnosis (categorical, categorized age measured at initial cancer diagnosis: <65, 65+)
- dem_age_le_18_flag (logical, indicating if patients was at least (larger/equal; le) 18 years of age at index date)
- dem_age_index_cont (continuous, age measured at index date; note: date of birth has only year-granularity, hence age is imprecise)
- dem_age_index (nominal, categorized age measured at index date: <65, 65+)
- dem_sex (binary, Male, Female)
- dem_race (nominal)
- dem_ethnicity (binary)
- dem_race_ethnicity (categorical, classification into five mutually exclusive groups according to SEER
- dem_state (character, US state of the center/network the patient is receiving care at)
- dem_region (nominal, region of the center/network the patient is receiving care at, can be Midwest, Northeast, South, West)
- dem_practice (nominal, setting patient is receiving care at, i.e. Academic, Community or both)
- dem_ses (nominal, socioeconomic status (SES) index based on residence area of patient; can be from '1 - Lowest SES' through '5 - Highest SES)
- c_smoking_history (logical, history of smoking; TRUE = History of smoking, FALSE = No history of smoking)

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb1_get_demographics(
    index_date = "dt_index",
    cancer = "aNSCLC"
    )

## End(Not run)
```

`edb1_get_diagnosis_heme` *Query diagnostic details for heme tumors in EDB1 database*

Description

Function queries diagnosis details including ISS staging information.

Usage

```
edb1_get_diagnosis_heme(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = "MultipleMyeloma"
)
```

Arguments

`x`	dataframe queried from edb1 with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB1 data/files are located
`cancer`	character, so "MultipleMyeloma"

Details

Function queries diagnosis details for the index cancer in EDB1 for all patients included in `x`. For patients to be eligible to be sampled in EDB1, they require to be diagnosed with multiple myeloma (identified via respective ICD-9 or ICD-10 codes in structured data) and have at least two documented clinical visits, on different days, occurring on or after January 1, 2011. Further criteria are subsequently assessed by processing of unstructured data to verify a true cancer diagnosis. Those criteria will be described in the protocols for the respective cancer-specific emulations. Most, if not all, data for a patients are captured after the initial diagnosis date which means that a potential treatment for this cancer is always after the initial diagnosis date.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort `x`.
2. For each patient the date of initial cancer diagnosis is provided; this date is captured from unstructured data.
3. The time from initial cancer diagnosis to the index date is computed as `index_date` - `dt_initial_dx`.
4. If the ISS stage is available:
   - The stage is captured from the structured `issstage` variable by removing the "Stage" prefix. "Unknowns" are mapped to missing stage values.
   - Relevant multiple myeloma-specific proteins (M-protein IgG class, free light chain kappa involvement, free light chain lambda involvement) are categorized as boolean TRUE/FALSE. Missings are mapped to values with "not documented".
5. Experimental: The number and location of potential secondary malignancies is inferred by C77.x, C78.x and C79.x ICD-10 codes (and corresponding ICD-9 mappings) from EDB1's diagnosis table. The `c_number_met_sites` is inferred by counting the unique number of sites as given by a C77.x, C78.x, C79.x granularity in the secondary diagnosis table. For more see `icd_metastases` system file. The resulting `c_number_met_sites` and `c_met_sites` are highly experimental variables and should be used with caution.
6. The newly created variables are joined back to `x`.

Value

x with all additional diagnosis variables joined, that is:

- `dt_initial_dx` (date, date of diagnosis or first documented diagnosis date for tumor)
- `c_stage_initial_dx` (nominal, summary ISS stage at initial diagnosis)
- `c_time_dx_to_index` (continuous, time between initial diagnosis and index date (in days))
- `c_m_protein_igg` (logical, whether the patient's immunoglobulin class of M protein is IgG)
- `c_light_chain_kappa` (logical, whether the patient's involved light chain is Kappa)
- `c_light_chain_lambda` (logical, Whether the patient's involved light chain is Lambda)

Experimental:

- `c_number_met_sites` (integer, number of metastatic sites for a given patient anytime before/on index date (inferred from ICD codes, see `icd_metastases` system file)
- `c_met_sites` (character string, description of anatomical locations of metastatic sites for a given patient patient's index date)

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_diagnosis_heme(
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = "MultipleMyeloma"
  )


## End(Not run)
```

`edb1_get_diagnosis_solid` *Query initial and metastatic diagnosis details for EDB1 database*

Description

Function queries diagnosis details including staging information.

Usage

```
edb1_get_diagnosis_solid(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL
)
```

Arguments

`x`	dataframe queried from edb1 with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB1 data/files are located
`cancer`	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC" or "MetastaticBreast"

Details

Function queries diagnosis details for the index cancer in EDB1 for all patients included in `x`. For patients to be eligible to be sampled in EDB1, they require to be diagnosed with lung, breast, or colorectal cancer (identified via respective ICD-9 or ICD-10 codes in structured data) and have at least two documented clinical visits, on different days, occurring on or after January 1, 2011. Further criteria are subsequently assessed by processing of unstructured data to verify a true cancer diagnosis. Most, if not all, data for a patients are captured after the initial diagnosis date which means that a potential treatment for this cancer is always after the initial diagnosis date.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort `x`.
2. For each patient the date of initial cancer diagnosis is provided and converted to YMD format; this date is captured from unstructured data and is typically the earliest date of the first pathology/cytology procedure that confirmed the malignant diagnosis.
3. The time from initial cancer diagnosis to the index date is computed as `index_date - dt_initial_dx`.
4. If recorded for a given cancer, the disease site is returned as `c_diseasesite` as provided in the table.
5. If a summary group stage is recorded for a given cancer, this information returned as `c_stage_initial_dx` as provided in the table. The stage is hierarchically determined via the pathologic group stage, clinical group stage or from a mapping from TNM to the most relevant version of AJCC. If the value equals any of "Not documented", "Occult", "Unknown", "Group stage is not reported", a missing value is returned.

6. If both a pathological and clinical summary group staging information are available (early NSCLC and early BC), the pathological group stage is prioritized before the clinical group stage. If the value equals any of "NA - Patient had pCR", or "Unknown/not documented, a missing value is returned.
7. If a metastatic diagnosis date is generally captured for the index cancer:
   - The metastatic diagnosis date is returned, that is, the date a patient was found to be metastatic (stage IV). This could either coincide with the initial diagnosis date or is determined hierarchically via the biopsy specimen collection date of the first metastasis from the pathology report, physician-reported date of biopsy, date of the radiology scan that indicates metastatic disease (if later confirmed by the treating physician), physician-reported date of metastatic diagnosis or the date of definitive surgery performed to metastatic site. For aNSCLC, the metastatic diagnosis date is extracted from unstructured documents using natural language processing if a patient was not directly stage IV at initial diagnosis.
   - The time from metastatic cancer diagnosis to the index date is computed as `index_date - dt_met_dx`.
   - A boolean de novo metastatic status (`c_de_novo_mets_dx`) is derived hierarchically and is returned as
     - TRUE if a patient is diagnosed in stage IV, if the metastatic diagnosis date is on or precedes the initial diagnosis date
     - a missing value (NA) if both the stage at initial diagnosis and the date of metastatic diagnosis are missing
     - FALSE if none of the above apply
   - A boolean `c_met_pre_index` variable is derived which indicates if there is evidence of a metastasis at anytime before or on the index date (which includes de novo metastatic patients AND progressors). This is derived hierarchically and is returned as
     - TRUE if the metastatic diagnosis date coincidences or precedes the index date or if the patient has a de novo metastatic status
     - a missing value (NA) if both the summary group stage at initial diagnosis and the date of metastatic diagnosis are missing
     - FALSE if none of the above apply
8. If an advanced diagnosis (i.e. stage IIIB-IV) date is recorded for a given cancer, the time from metastatic cancer diagnosis to the index date is computed as `index_date` - advanced diagnosis date.
9. Experimental: The number and location of potential secondary malignancies is inferred by C77.x, C78.x and C79.x ICD-10 codes (and corresponding ICD-9 mappings) from EDB1's diagnosis table. The `c_number_met_sites` is inferred by counting the unique number of sites as given by a C77.x, C78.x, C79.x granularity in the secondary diagnosis table. For more see `icd_metastases` system file. The resulting `c_number_met_sites` and `c_met_sites` are highly experimental variables and should be used with caution.
10. The newly created variables are joined back to `x`.

Value

x with all additional diagnosis variables joined, that is:

- `dt_initial_dx` (date, date of diagnosis or first documented diagnosis date for tumor)
- `c_stage_initial_dx` (nominal, summary group stage at initial diagnosis, if available)
- `dt_met_dx` (date, date of earliest evidence of distant metastasis)

- c_de_novo_mets_dx (binary logical, evidence of presence of one or multiple metastases at/before initial diagnosis)
- c_time_dx_to_index (continuous, time between initial diagnosis and index date (in days))
- c_time_adv_dx_to_index (continuous, time between advanced diagnosis and index date (in days; advanced NSCLC only))
- c_time_met_dx_to_index (continuous, time between earliest evidence of a metastatic diagnosis and index date (in days; not in early NSCLC))
- c_met_pre_index (binary logical, evidence of any metastasis before/on index date; includes de novo metastatic patients and progressors (overlap with c_de_novo_mets_dx possible))

Experimental:

- c_number_met_sites (integer, number of metastatic sites for a given patient anytime before/on index date (inferred from ICD codes, see icd_metastases system file)
- c_met_sites (character string, description of anatomical locations of metastatic sites for a given patient patient's index date)

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_diagnosis_solid(
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = "aNSCLC"
  )


## End(Not run)
```

---

edb1_get_ecog *Query performance status information for an inception cohort*

Description

Function queries ECOG performance status tables and curates derived variables

Usage

```
edb1_get_ecog(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

x	dataframe queried with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date
path	character string, path to directory where EDB1 data/files are located
cancer	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"
from	integer, left boundary of ECOG measurement window relative to index date (e.g., -90, indicating ECOG should be measured not before 90 days before index date)
to	integer, right boundary of ECOG measurement window relative to index date (e.g., 0, indicating ECOG should be measured until day of index date (inclusive))
ties	character, one of "lower" or "higher" to choose either the lower (default) or higher ECOG measurement if there are two measurements on the same day
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function considers both NLP-extracted and structured ECOG measurements (to enhance the availability of ECOG measurements). All ECOG measurements are identified within the baseline measurement window, then the closest measurement relative to the index date is selected. If there are two measurements within the same closest distance, the lower (default) or higher (depending on how `ties` is specified) is prioritized.

The function operates via the following logic:

1. Pull ECOG measurements from structured sources and NLP-derived ECOG measurements and subset to patients included in inception cohort `x`.
2. Blend ECOG measurements from both sources, subset to measurements within the specified measurement window (via `from` and `to`) relative to `index_date` and discard measurements with missing ECOG value.
3. Prioritize ECOG measurements assessed in closest absolute distance to the `index_date`.
   - if `ties` = `lower`: for each patient, arrange ECOG measurement by ascending absolute distance to index and ascending ECOG value (= lowest value) and pick the first observation
   - if `ties` = `higher`: for each patient, arrange ECOG measurement by ascending absolute distance to index and descending ECOG value (= highest value) and pick the first observation
4. The newly created variables are joined back to `x`.

Value

x with all additional ECOG variables joined, that is:

- `c_ecog_{from}_{to}`, ECOG value measured in the specified measurement window
- `c_ecog_{from}_{to}`, distance of the date the ECOG value was measured relative to the index date

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_ecog(
    cancer = "aNSCLC",
    from = -180,
    to = 0,
    ties = "lower"
    )

## End(Not run)
```

`edb1_get_histology` *Query histology information for an inception cohort*

Description

Function queries histology information for a solid tumor inception cohort.

Usage

```
edb1_get_histology(
  x = NULL,
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  histology_match = NULL
)
```

Arguments

`x`	dataframe with at least patient ids and index date of inception cohort
`path`	character string, path to directory where EDB1 data/files are located
`cancer`	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"
`histology_match`	character, string match to categorize and identify patients with a certain histology for the indicated tumor site, e.g. "non-squamous cell carcinoma".

Details

The function queries histology measurements if available for a given cancer. In EDB1, there is one unique histology measurement per patient abstracted from pathology or clinical reports without indication of a specific date. In general, the histology for solid tumors is typically assessed around the time of diagnosis or surgery.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort `x`.

2. Create a binary boolean variable (TRUE/FALSE) indicating if the histology description/text string matches the keyword string in `histology_match`, e.g., a patient with a "squamous cell carcinoma" histology would return TRUE if "squamous" was the `histology_match` keyword string. The keyword search string defined in argument `histology_match` is not case sensitive and missing measurements are discarded. The name of the variable is adapted to match the keyword string.
3. The actual histology string is propagated as recorded in the database and returned as `c_histology`.
4. The newly created variables are joined back to `x`.

Note: Some patients may have more than one histology recording across cohorts since they can have multiple primaries. However, this function is designed to query histology information for one cancer type at a time. That means, there is just one recording per patient.

Value

x with all additional histology variables joined, prepended with "c_"

- `c_histology` (nominal, histology recorded for given patient)
- `c_(histology_match)` (logical, there is a string match for histology specified by `histology_match`. `FALSE` may also include "unknowns" or "NOS" whose information was just not granular enough to be able to determine the histological subtype with absolute certainty

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_histology(
    cancer = "aNSCLC",
    histology_match = "Non-squamous cell carcinoma"
    )

## End(Not run)
```

---

edb1_get_labs *Query lab information for a given inception cohort*

Description

Function queries the lab table and standardizes according to a reference measurement unit.

Usage

```
edb1_get_labs(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  lab_name = NULL,
  from = -90,
```

```
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB1 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB1 data/files are located
cancer	character, "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"
lab_name	character, curated name of lab (see details)
from	integer, left boundary of lab measurement window relative to index date (e.g., -90, indicating lab should be measured not before 90 days before index date)
to	integer, right boundary of lab measurement window relative to index date (e.g., 0, indicating lab should be measured not later than the day of the index date (inclusive))
ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" (default) lab measurement be prioritized
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA? Lower and upper thresholds are documented in labs_mapping_edb1
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function queries and cleans supported lab measurements. In detail, the function selects measurements in a from - to measurement window relative to the index date. The date of the lab test is derived as the earliest of the test date or result date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the lower (default) or higher measurement can be prioritized.

EDB1 already comes with standardized test units which are used as reference units for the selected lab tests across all databases used in this project.

Note: Only selected labs are supported by this function which were taken from the ROPRO prognostic score (Becker T et al., Ann Oncol 2020).

The supported and available labs are:

- c_albumin_g_l (albumin mass/volume in serum or plasma)
- c_alp_u_l (alkaline phosphatase enzymatic activity/volume in serum or plasma)
- c_alt_u_l (alanine aminotransferase enzymatic activity/volume in serum or plasma)
- c_ast_u_l (aspartate aminotransferase enzymatic activity/volume in serum or plasma; used to compute ast-alt ratio)

- c_bilirubin_mg_dl (total bilirubin mass/volume in serum or plasma)
- c_calcium_mg_dl (calcium mass/volume in serum or plasma)
- c_chloride_mmol_l (chloride moles/volume in serum or plasma)
- c_eosinophils_leukocytes_ratio (eosinophils/100 leukocytes in blood)
- c_glucose_mg_dl (glucose mass/volume in serum or plasma)
- c_granulocytes_leukocytes_ratio (granulocytes/100 leukocytes in blood)
- c_hemoglobin_g_dl (hemoglobin mass/volume in blood)
- c_ldh_u_l (lactate dehydrogenase enzymatic activity/volume in serum or plasma)
- c_lymphocyte_10_9_l (lymphocytes #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_lymphocyte_leukocyte_ratio (lymphocytes/100 leukocytes in blood)
- c_monocytes_10_9_l (monocytes #/volume in blood)
- c_neutrophil_10_9_l (neutrophils #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_platelets_10_9_l (platelets #/volume in blood)
- c_protein_g_l (protein mass/volume in Serum or Plasma)
- c_urea_nitrogen_mg_dl (urea nitrogen mass/volume in serum or plasma)

The function operates via the following logic:

1. Pull lab table and subset to patients included in inception cohort x.
2. Subset to lab of interest.
3. Measurements with missing lab test result or unit are discarded.
4. Lab test results are categorized into "normal" and "abnormal" depending on if they fall within the provided reference.
5. if set_implausible_na is set TRUE:
    - a lab test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the labs_mapping_implausible_values table.
6. The lab date is derived as the earliest of the test or result date.
7. All lab dates are restricted to the ones that fall within the measurement window specified in the from and to arguments for the lower and upper limits of the measurement window, respectively.
8. If multiple lab tests are available for a given patient, the prioritization step is carried out as follows:
    - For each patient, lab tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if ties = "higher") or ascending quantitative results to prioritize the lower measurement (if ties = "lower") measurements. The default is "lower".
    - The first row for each patient after the above sorting is selected
9. The newly created variables are named according to the standardized lab name and unit of measurement and joined back to x.

Value

x with all additional labs variables joined, that is:

- `c_(lab_name)_distance_(from)_(to)` - days from date of lab measurements to index date
- `c_(lab_name)_(unit)_(from)_(to)` - binary lab result indicating if lab was within ("normal") the reference range or outside ("abnormal")
- `c_(lab_name)_(unit)_(from)_(to)_cont` - quantitative lab result after unit harmonization/conversion

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_labs(
    cancer = "aNSCLC",
    lab_name = "c_albumin_g_l",
    set_implausible_na = TRUE,
    verbose = TRUE
  )


## End(Not run)
```

---

edb1_get_os *Query overall survival outcome for a given inception cohort*

Description

Function queries mortality and other information to derive a righ-censored time to all-cause mortality endpoint

Usage

```
edb1_get_os(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  data_cut_off_date = lubridate::ymd("2024-04-30"),
  verbose = TRUE
)
```

Arguments

x	dataframe queried from edb1 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where data/files are located
cancer	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"
data_cut_off_date	date of database lock; the data cut-off for this delivery (May 30, 2024) is April 30, 2024. This parameter can be changed if a grace period of x months before database lock is desired.
verbose	logical, print query progress and informative meta information

Details

The function queries and curates intention-to-treat (ITT) overall survival endpoint. The ITT follow up time is defined as the time from index date (dt_index, including the index date) to date of death. If a patient did not decease during follow-up, the patient will be censored at the last observed clinical activity (including visits and treatment information) or data cut-off date whichever is earlier.

In EDB1, the granularity of the date of death variable is given as month-year and (in rare cases) year only. In these cases, the date of death is imputed to the mid/15th of the month and the mid/July 2 of the year, respectively. This can lead to negative/implausible follow-up times if the index date is after the imputed date of death.

The function operates via the following logic:

1. Pull mortality details from the one-row-per-patient mortality table and subset to patients included in inception cohort x.
2. A flag is created for patient with year-level granularity in the date of death.
3. The date of death is imputed to the mid/15th of the month and the mid/July 2 of the year for patient with month-level (default) and year-level granularity, respectively.
4. A patient's last structured activity date is determined based on the last observed visit or systemic treatment (oral or administered).
5. Mortality and last structured activity dates are joined to x and a binary death event variable is derived based on if a date of death is captured for a given patient.
6. The censoring date is determined as the date of death, if available. If no death date is available, the censoring date is determined as the last structured activity (see step 4) or at the data cut-off date (2024-04-30 is the default), whichever is earlier.
7. The follow-up (FU) time, defined as the time from index date (including index date) until censor date, is derived in days, months (= FU time in days/30.417) and years, depending on what the time scale should be for the respective analysis.
8. The newly created variables are joined back to x.

Value

x with all endpoint information joined, that is:

- death_itt (binary, event indicator for all-cause mortality)
- fu_itt_days (numeric, ITT follow-up time in days)
- fu_itt_months, (numeric, ITT follow-up time in months (i.e.,fu_itt_days / 30.417))
- fu_itt_years, (numeric, ITT follow-up time in years)

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_os(
    cancer = "aNSCLC"
    )

## End(Not run)
```

`edb1_get_vitals` *Query vital sign measurements for a given cohort*

Description

Function queries vitals sign measurements

Usage

```
edb1_get_vitals(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  vital_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

`x`	dataframe queried from EDB1 with at least patient ids and therapy index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB1 data/files are located
`cancer`	character, "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"
`vital_name`	character, curated name of vital sign (see details)
`from`	integer, left boundary of vitals measurement window relative to index date (e.g., -90, indicating vital test should be measured not before 90 days before index date)
`to`	integer, right boundary of vitals measurement window relative to index date (e.g., 0, indicating vital test should be measured not later than the day of the index date (inclusive))

ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" vital test measurement be prioritized
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in vitals_mapping_edb1
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function queries and cleans all available vital sign measurements. In detail, the function removes measurements that are character strings and only considers quantitative results. In EDB1, unit-cleaned vital sign measurements are provided. However, due to frequent missing units, unit-cleaned vital sign measurements can exhibit high missingness. To mitigate this missingness, the function also considers "raw" vital sign measurements if no unit-cleaned measurement is observed. Hence, it is recommended to set set_implausible_na to TRUE to remove implausible values. The function further only selects measurements in a from - to measurement window relative to the index date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the lower (default) or higher measurement can be prioritized.

Note: Body mass index (BMII) and heart rate (HR) does not come with a unit-cleaned test result and the raw vital result is returned assuming uniform units. Height is uniformly measured in cm and is converted to meters.

The available and supported vitals are:

- c_sbp (systolic blood pressure in mmHg)
- c_dbp (diastolic blood pressure in mmHg)
- c_bmi (body mass index in kg/m^2 directly measured)
- c_height (height in m)
- c_hr (heart rate/pulse in beats/min)
- c_oxygen (oxygen saturation; taken from O2 sat and pulse oximetry)
- c_weight (weight in kg)

The function operates via the following logic:

1. Pull vitals table and subset to patients included in inception cohort x.
2. Subset to vitals measurement of interest.
3. Measurements with missing vital test result or unit are discarded.
4. if set_implausible_na is set TRUE:
    - a vitals test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the vitals_mapping_implausible_values table.
5. The vitals test date is derived as the earliest of the test or result date.
6. All vital dates are restricted to the ones that fall within the measurement window specified in the from and to arguments for the lower and upper limits of the measurement window, respectively.
7. If multiple vitals tests are available for a given patient, the prioritization step is carried out as follows:

- For each patient, vitals tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if `ties` = "higher") or ascending quantitative results to prioritize the lower measurement (if `ties` = "lower") measurements. The default is "lower".
- The first row for each patient after the above sorting is selected

8. The newly created variables are named according to the standardized vitals name and unit of measurement and joined back to `x`.

Value

x with all additional vital test variables joined, that is:

- `c_(vital_name)_distance_(from)_(to)` - days from date of vital sign measurements to index date
- `c_(vital_name)_(unit)_(from)_(to)_cont` - quantitative vital sign measurement

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb1_get_vitals(
    cancer = "aNSCLC",
    vital_name = "c_oxygen",
    set_implausible_na = TRUE,
    verbose = TRUE
  )


## End(Not run)
```

---

edb1_query_ropro *Query and curate all relevant ROPRO variables*

Description

This function queries, cleans and transforms all necessary covariates needed to compute the ROPRO prognostic score in EDB1.

Usage

```
edb1_query_ropro(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb1"),
  cancer = NULL,
  from = -90,
  to = 0,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB1 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index date, default is dt_index
path	character string, path to directory where EDB1 data/files are located
cancer	character, one of "aNSCLC", "CRC", "EarlyBreast", "EarlyNSCLC", "MetastaticBreast" or "MultipleMyeloma"
from	integer, left boundary of measurement window for time-dependent variables relative to index date (e.g., -90, indicating variables should be measured not before 90 days before index date)
to	integer, right boundary of measurement window for time-dependent variables relative to index date (e.g., 0, indicating variables should be measured not later than the day of the index date (inclusive))
verbose	logical, print progress of query

Details

Wrapper around major functions to query required covariates to compute ROPRO. Selected covariates are log transformed or log-log transformed. More details, see Becker, Weberpals, et al., Ann Oncol 2020.

Value

x with all required ROPRO covariates joined. All general and cancer type-specific (as specified in cancer argument) covariates are returned.

Note that:

- there is no specific ROPRO for EarlyNSCLC, so only the covariates for the general pan-tumor ROPRO will be returned
- there is only a ROPRO for metastatic CRC (no early CRC)
- covariates for EarlyBreast are identical to the the general pan-tumor ROPRO, just the weights are different

Examples

```
## Not run:
library(encore.io)

x_ropro <- x |>
  edb1_query_ropro(
    index_date = "dt_index",
    path = Sys.getenv("path_edb1"),
    cancer = "aNSCLC",
    from = -90,
    to -0,
    verbose = TRUE
    )


## End(Not run)
```

edb2_assign_date *Helper function to assign an actual date for a xxx_timedelta variable in edb2*

Description

In the edb2 database, time differences (timedelta) are assigned for clinical events. These timedeltas are always relative to the initial cancer diagnosis for which a given patient sampled into the database. This function helps to assign actual dates based on on those time differences.

Usage

```
edb2_assign_date(
  x = NULL,
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC")
)
```

Arguments

x	dataframe queried from edb2 with at least the patient id column and columns that end on "timedelta"
path	string, path to edb2 root directory
cancer	character, either "NSCLC" or "MM"

Details

CAVEAT: both the date of initial diagnosis and other dates come with imprecision (_imp). There are 3 possibilities: 0: There is no imprecision. precise date (MM/DD/YYYY) associated with this event. 15: There is imprecision. imprecise date (MM/YYYY) associated with this event. 182: There is imprecision.imprecise date (YYYY) associated with this event.

Value

x including the date colum(s) of all timedelta columns (.col) with naming convention dt_{.col}"

Examples

```
## Not run:
library(encore.io)


## End(Not run)
```

edb2_compute_ropro *Derive ROPRO prognostic score*

Description

This function computes ROPRO prognostic score (Becker, Weberpals, et al., Ann Oncol 2020).

Usage

```
edb2_compute_ropro(x = NULL, cancer = c("NSCLC", "MM"))
```

Arguments

x	dataframe with inception cohort and required ROPRO covariates
cancer	character, what cancer-specific ROPRO should be computed ("NSCLC", "MM")

Details

This function takes in a dataframe with all required variables to compute the general and cancer-specific ROPRO (specified in cancer). The variables need to be queried and transformed before which can be done through th edb2_query_ropro functions (specific for each database).

Important: Since EDB2 does not have all required variables, please use the edb2_compute_ropro function before use of this function to compute a reduced ROPRO model.

Value

The function returns x with the final general and cancer-specific ROPRO

Examples

```
## Not run:
library(encore.io)

x_ropro <- x |>
  edb2_query_ropro(
    index_date = "dt_index",
    path = Sys.getenv("path_edb2"),
    cancer = "NSCLC",
    from = -90,
    to = 0,
    verbose = TRUE
    ) |>
 edb_2_compute_ropro(
    cancer = "NSCLC"
   )


## End(Not run)
```

`edb2_get_biomarker` *Query biomarker information for a given solid tumor inception cohort*

Description

Function queries biomarker tables and curates information on alterations in defined driver genes.

Usage

```
edb2_get_biomarker(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC"),
  biomarker_name = NULL,
  from = -90,
  to = 0,
  label_name = FALSE
)
```

Arguments

`x`	dataframe queried from edb2 with at least patient ids and therapy index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB2 data/files are located
`cancer`	character, one ofMM or NSCLC
`biomarker_name`	character, name of biomarker/gene alteration (see details)
`from`	integer, left boundary of biomarker measurement window relative to index date (e.g., -90, indicating biomarker should be measured not before 90 days before index date)
`to`	integer, right boundary of biomarker measurement window relative to index date (e.g., 0, indicating biomarker should be measured until day of index date (inclusive))
`label_name`	logical, should variable name carry information about the measurement window

Details

The function queries and categorizes a certain biomarker that was collected and curated as part of the database as either positive or negative. The categorization happens according to if the biomarker mutation/alteration is a clinically actionable one. For example, patients with any mutation in the EGFR gene or MSI-H/dMMR status would be classified as positive. In case there are multiple measurements per biomarker and patient, the time point of measurement is defined as the non-missing biomarker result in the covariate ascertainment window given by `from` and `to` that is closest to the `index_date`. The biomarker date (`dt_(biomarker)`) in this database is defined as the earliest of specimen collection, result report or documented date. There can be imprecisions to the date of measurement on the month or year granularity level.

Depending on the cancer type, the biomarker names can be: 1p, 1q, ALK, BRAF, Complex Cytogenetics/Karyotype, DDR2, del(13), del(17), del(17p), Diploid, EGFR, FGFR1, HER2 (ERBB2),

Hyperploid, Hypoploid, KEAP1 (INRF2), KRAS, MEK1(MAP2k1), MEK2 (MAP2K2), MET, MLH1, MMR, MSH2, MSH6, MSI/Microsatellite Instability, Normal Cytogenetics/Karyotype, NRAS, NTRK1, NTRK2, NTRK3, PD-L1, PIK3CA, PMS2, RET, ROS1, STK11 (LKB1), t(11;14), t(14;16), t(14;20), t(4;14), t(6;14), TMB/Tumor Mutational Burden, TP53

The function operates via the following logic:

1. Pull biomarker table and subset to patients included in inception cohort `x`.
2. Subset table to biomarker names as specified in `biomarker_name`.
3. Biomarker status test result is categorized hierarchically:
   - If biomarker interpretation is any of "negative" OR "no_loss_of_expression" OR "low" OR "intermediate" OR "stable" OR "proficient", the result is mapped to "negative".
   - If biomarker interpretation is any of "positive" OR "loss_of_expression" OR "high" OR "deficient" OR "unstable", the result is mapped to "positive".
   - If none of the above applies, the result is mapped to NA (missing).
4. Missing mapped biomarker status results are discarded.
5. The biomarker date is specified as the earliest of the collected date, reported date or documented date.
6. Biomarker dates outside of the specified measurement window relative to the index date (defined using the `index_date`, `from` and `to` arguments) are discarded.
7. If multiple biomarker tests are available for a given patient, the prioritization step is carried out as follows:
   - For each patient, biomarker mappings (negative, positive) and the absolute distance from biomarker date to index date are sorted in descending and ascending order, respectively.
   - To prioritize any positive biomarker mapping closest to the index date, the first row for each patient is selected. This reflects the final mapped biomarker status variable.
   - In addition, all available biomarker details within the measurement window are collapsed into a new variable. This reflects the final biomarker detail variable.
8. The newly created biomarker variables are named accordingly and returned.
9. The newly created variables are joined back to `x`.

WARNING: the categorization of more complex biomarkers like HER2 requires a more refined approach or de novo written code to account for equivocal results.

Value

x with all additional biomarker variables joined, that is:

- `c_(biomarker_name)_status_(from)_(to)` (binary, biomarker mutation status positive or negative)
- `c_(biomarker_name)_detail_(from)_(to)` (character string, more details about selected measurement)
- `c_(biomarker_name)_detail_all_(from)_(to)` (character string, this provides details about all results and details for this biomarker if there were multiple tests in the measurement window (from, to))
- `c_(biomarker_name)_distance_(from)_(to)` (numeric, relative distance between date of measurement and index date in days)
- `dt_(biomarker_name)_(from)_(to)` (date, date of selected biomarker measurement)

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb2_get_biomarker(
    cancer = "NSCLC",
    biomarker_name = "egfr",
    from = -180,
    to = 0)

## End(Not run)
```

`edb2_get_demographics` *Query demographic variables for an inception cohort*

Description

Function queries all available demographic variables from the EDB2 database, curates them and joins them to the inception cohort x.

Usage

```
edb2_get_demographics(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC")
)
```

Arguments

`x`	dataframe queried from edb2 with at leat patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB2 data/files are located
`cancer`	character, one of MM or NSCLC

Details

The function queries and categorizes demographic variables as reported in the electronic health record (EHR). The function operates via the following logic:

Demographics:

1. Pull the initial diagnosis date of the primary cancer and subset to patients included in inception cohort x.
2. Pull the demographics table and join the diagnosis date and index date for each patient.
3. The patient age at initial diagnosis is provided but truncated (missing) for patients of age >89 at initial diagnosis.

4. The index date is derived as age at diagnosis plus the time delta (in years) that is associated with the treatment start (= index) date.
5. Additional categorical age variables are computed as <65 and 65+ years of age.
6. The reported sex, race and ethnicity variables are provided in a one-row-per-patient format and directly mapped one-to-one. If race is categorized as "Other", it is set to missing.
7. Additionally, race and ethnicity are combined into a new combined variable `dem_race_ethnicity` with five mutually exclusive groups as defined by SEER.
8. The newly created variables are joined back to `x`.

Smoking: There may be multiple recording for history of tobacco use. The function uses the earliest of the assessed or documented date of tobacco use before or on the index date. If there are multiple records for history of tobacco use, the function prioritizes any evidence of smoking (1st priority) and the most recent recording (2nd priority) before or on the index date. The database provides the type of tobacco product which the function does not distinguish.

Value

x with all additional demographic variables joined (see details), that is,

- dem_age_initial_diagnosis (categorized age measured at initial cancer diagnosis: <60, 60-69, 70-79, 80+)
- dem_age_le_18_flag (logical indicating if patients was at least (larger/equal; le) 18 years of age at index date)
- dem_age_index_cont (continuous age measured at index date, CAVE: edb2 truncates age to 89 years for all patients >89, which means that `dem_age_index_cont` will show NA for these patients)
- dem_age_index (categorized age measured at index date: <60, 60-69, 70-79, 80+)
- dem_sex (binary, Male, Female, NA)
- dem_race (categorical, "", "Declined" and "Other" are converted to NA)
- dem_ethnicity (binary, "" and "Declined" are converted to NA)
- dem_race_ethnicity (categorical, classification into five mutually exclusive groups according to SEER)
- c_smoking_history (binary, history of any tobacco use on or before index date, TRUE = yes, FALSE = no)

Examples

```
## Not run:
library(encore.io)
ard <- x |>
  edb2_get_demographics(
    index_date = "dt_index",
     cancer = "BC"
     )

## End(Not run)
```

edb2_get_diagnosis_heme *Query diagnostic details for heme tumors in EDB2 database*

Description

Function queries diagnosis details including ISS staging information.

Usage

```
edb2_get_diagnosis_heme(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = "MultipleMyeloma"
)
```

Arguments

x	dataframe queried from edb2 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB2 data/files are located
cancer	character, so "MultipleMyeloma"

Details

Function queries diagnosis details for the index cancer in EDB2 for all patients included in x. For a pathologically confirmed diagnosis, the date of collection reported on the earliest pathology report confirming the malignancy is used as the initial diagnosis date. When a patient is clinically diagnosed and treated without pathological confirmation, the date of the imaging procedure, lab test, or clinical assessment cited by the oncologist as identifying the diagnosis is captured as the date of initial diagnosis. The detailed criteria for sampling eligibility will be described in the protocols for the respective cancer-specific emulations.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort x.
2. For each patient the date of initial cancer diagnosis is provided (see details above).
3. The time from initial cancer diagnosis to the index date is computed as index_date - dt_initial_dx.
4. Multiple myeloma staging is based on ISS or R-ISS (modified ISS). The staging method is likely depended on the diagnosis year and if mutliple are available for a given patient R-ISS is prioritized over ISS. If the method is "Unknown", that staging is used with the lowest priority.
5. Relevant multiple myeloma-specific proteins (M-protein IgG class, free light chain kappa involvement, free light chain lambda involvement) are categorized as present/not present.
6. Experimental: the number c_number_met_sites and location c_met_sites of historical secondary malignancies is inferred from a patients' cancer history at the time of diagnosis. This includes cancer diagnoses that are no longer active at the time of the current cancer diagnosis and where all treatment has been completed.

7. The newly created variables are joined back to x.

Note: all times/dates can be associated with a level of imprecision on the month-level or year-level.

Value

x with all additional diagnosis variables joined, that is:

- dt_initial_dx (date, date of diagnosis or first documented diagnosis date for tumor)
- c_stage_initial_dx (nominal, summary ISSS stage at initial diagnosis)
- c_time_dx_to_index (continuous, time between initial diagnosis and index date (in days))
- c_m_protein_igg (logical, whether the patient's immunoglobulin class of M protein is IgG)
- c_light_chain_kappa (logical, whether the patient's involved light chain is Kappa)
- c_light_chain_lambda (logical, Whether the patient's involved light chain is Lambda)

Experimental:

- c_number_met_sites (integer, number of metastatic sites for a given patient anytime before/on index date (inferred from ICD codes, see icd_metastases system file)
- c_met_sites (character string, description of anatomical locations of metastatic sites for a given patient patient's index date values are separated by ':,')

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb2_get_diagnosis_heme(
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = "MultipleMyeloma"
  )


## End(Not run)
```

---

edb2_get_diagnosis_solid

Query initial and metastatic diagnosis dates for EDB2 database

---

Description

Function queries diagnosis details including staging information.

Usage

```
edb2_get_diagnosis_solid(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = "NSCLC"
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, NSCLC (default)

Details

Function queries diagnosis details for the index cancer in EDB2 for all patients included in x. For a pathologically confirmed diagnosis, the date of collection reported on the earliest pathology report confirming the malignancy is used as the initial diagnosis date. When a patient is clinically diagnosed and treated without pathological confirmation, the date of the imaging procedure, lab test, or clinical assessment cited by the oncologist as identifying the diagnosis is captured as the date of initial diagnosis. The detailed criteria for sampling eligibility will be described in the protocols for the respective cancer-specific emulations.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort x.
2. For each patient the date of initial cancer diagnosis is provided (see details above).
3. If a summary group stage is recorded for a given cancer, this information returned as c_stage_initial_dx as provided in the table (see details above). If the value equals any of "unspecified", "discrepant_information", "discrepant information", a missing value is returned.
4. If the time of AJCC staging is provided as either the assessed or documented time, the earlier of the two is used as the date of AJCC staging (which is different than the date of initial diagnosis).
5. Patients without staging information are discarded.
6. If both pathological and clinical staging is available, pathological staging is prioritized.
7. Steps 1.-7. are repeated for additional M staging details (= presence of metastases) coming from the TNM stage table.
8. Details on distant metastases are pulled from the metastatic sites tables.
9. The date of each metastatic site recordings is consolidated as the earliest of the assessed and documented time.
10. Restrict all metastasis measurements to time before or on the index date.
11. For each patient, the sites and number of sites is summarized. To that end, each row (= recording) per patient is counted as a unique metastatic site.
12. The date of the earliest evidence of a distant metastasis (dt_met_dx) is derived as the earliest recording of a metastatic site (see step 10) before or on index date.
13.
    - A boolean de novo metastatic status (c_de_novo_mets_dx) is derived hierarchically and is returned as
      - TRUE if a patient is diagnosed in stage IV, if the M stage equals 1, if the metastatic diagnosis date is on or precedes the initial diagnosis date
      - a missing value (NA) if both the stage at initial diagnosis, the M stage and the date of metastatic diagnosis are missing
      - FALSE if none of the above apply
14. The time from initial cancer diagnosis to the index date is computed as index_date - dt_initial_dx.

15. The time from metastatic cancer diagnosis to the index date is computed as `index_date` - `dt_met_dx`.
16. A boolean `c_met_pre_index` variable is derived which indicates if there is evidence of a metastasis at anytime before or on the index date (which includes de novo metastatic patients AND progressors). This is derived hierarchically and is returned as
    - TRUE if the metastatic diagnosis date coincidences or precedes the index date of if the patient has a de novo metastatic status
    - a missing value (NA) if both the summary group stage at initial diagnosis, the date of AJCC staging and the date of M staging are missing
    - FALSE if none of the above apply
17. The newly created variables are joined back to `x`.

Note: all times/dates can be associated with a level of imprecision on the month-level or year-level.

Value

x with all additional diagnosis variables joined, that is:

- `dt_initial_dx` - Date of diagnosis or first documented diagnosis date for tumor (de-identified to week)
- `dt_staging` - Date stage was recorded, if available (de-identified to week)
- `dt_met_dx` - Date of earliest evidence of distant metastasis (de-identified to week)
- `c_stage_initial_dx` - First summary group stage at initial diagnosis, if available
- `c_tnm_initial_dx` - Individual TNM staging values/stages at initial diagnosis
- `c_de_novo_mets_dx` - Evidence of presence of one or multiple metastases at/before initial diagnosis
- `c_time_dx_to_index` - Time between initial diagnosis and index date (in days)
- `c_time_met_dx_to_index` - Time between earliest evidence of a metastatic diagnosis and index date (in days)
- `c_met_pre_index` - Evidence of any metastasis between initial diagnosis and index date (logical); includes initial diagnosis date (overlap with c_de_novo_mets_dx possible)
- `c_number_met_sites` - number of metastatic sites for a given patient anytime before/on index date (inferred from provided metastatic site description)
- `c_met_sites` - description of anatomical locations of metastatic sites for a given patient patient's `c_number_met_sites`

Examples

```
## Not run:
library(encore.io)
analysis_cohort <- x |>
 edb4_get_diagnosis(cancer = "NSCLC")


## End(Not run)
```

edb2_get_ecog *Query performance status information for a given cohort*

Description

Function queries performance status (ECOG, Karnofsky) tables and curates derived variables

Usage

```
edb2_get_ecog(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC"),
  from = -90,
  to = 0,
  ties = "lower",
  label_name = FALSE
)
```

Arguments

x	dataframe queried from edb2 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of MM or NSCLC
from	integer, left boundary of ECOG measurement window relative to index date (e.g., -90, indicating ECOG should be measured not before 90 days before index date)
to	integer, right boundary of ECOG measurement window relative to index date (e.g., 0, indicating ECOG should be measured until day of index date (inclusive))
ties	character, one of "lower" or "higher" to choose either the lower (default) or higher ECOG measurement if there are two measurements on the same day
label_name	logical, should variable name carry information about the measurement window #treat character, column indicating binary exposure status (needed for measurement summary statistics by treatment status)

Details

The function queries all ECOG and Karnofsky measurements, then maps the Karnofsky measurment to an ECOG value according to Oken et al. (Am J Clin Oncol 1982), then filters for all measurements within the indicated time window specified by `from` and `to`. It then chooses the measurement closest to the index date (closest relative distance). In case of ties, the user has the option to choose the lower or higher (`ties`) measurement.

The function operates via the following logic:

1. Pull performance score measurements from structured sources and subset to patients included in inception cohort `x`.
2. Define the performance score date as the earliest of the performance score reported or documented timedelta relative to the `index_date`.
3. In rare cases, there are performances scores >5 that are erroneously labelled as ECOG and performance scores with values between 1-5 that are labelled as Karnofsky scores; these labels get cleaned to the respective source of the performance score.
4. Karnofsky performance scores are mapped to their ECOG score equivalents as per Oken et al. (Am J Clin Oncol 1982).
5. Subset to ECOG measurements within the specified measurement window (via `from` and `to`) relative to `index_date` and discard measurements with missing ECOG value.
6. Prioritize ECOG measurements assessed in closest absolute distance to the `index_date`.
    - if `ties = lower`: for each patient, arrange ECOG measurement by ascending absolute distance to index and ascending ECOG value (= lowest value) and pick the first observation
    - if `ties = higher`: for each patient, arrange ECOG measurement by ascending absolute distance to index and descending ECOG value (= highest value) and pick the first observation
7. The newly created variables are joined back to `x`.

Value

x with all additional ECOG variables joined, that is:

- `c_ecog_{from}_{to}`, ECOG value measured in the specified measurement window
- `c_ecog_{from}_{to}`, distance of the date the ECOG value was measured relative to the index date

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb2_get_performance(
    cancer = "NSCLC",
    from = -180,
    to = 0,
    ties = "lower"
    )

## End(Not run)
```

edb2_get_histology *Query histology information for an inception cohort*

Description

Function queries histology information for a solid tumor inception cohort.

Usage

```
edb2_get_histology(
  x = NULL,
  path = Sys.getenv("path_edb2"),
  cancer = "NSCLC",
  histology_match = NULL,
  return_all = FALSE
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
path	character string, path to directory where EDB2 data/files are located
cancer	character, NSCLC
histology_match	character, string match to categorize and identify patients with a certain histology for the indicated tumor site, e.g. "adenocarcinoma"
return_all	logical, should a variable be returned that summarizes all recorded histology measurements? default is FALSE

Details

The function queries histology measurements if available for a given cancer. In EDB2, many patients can have more than one histological subtype recorded without indication of a specific date. The function prioritizes histology measurements that match the keyword search string in histology_match.

The function operates via the following logic:

1. Pull histology recording details from the histology table and subset to patients included in inception cohort x.
2. Create a binary boolean variable (TRUE/FALSE) indicating if the histology description/text string matches the keyword string in histology_match, e.g., a patient with a "squamous cell carcinoma" histology would return TRUE if "squamous" was the histology_match keyword string. The keyword search string defined in argument histology_match is not case sensitive and missing measurements are discarded. The name of the variable is adapted to match the keyword string.
3. To receive one measurement per patient, histology measurements are prioritized that match the keyword search string in histology_match (i.e., return TRUE).
4. All available histology measurement for a given patient are summarized and returned in c_histology_all if return_all is specified as TRUE.

5. The newly created variables are joined back to x.

Tip: the function can also be stacked/executed multiple times with different histological subtypes.

Value

x with all additional histology variables joined, prepended with "c_"

- c_histology_match): A TRUE/FALSE if the histological subtype was observed for a given patient. FALSE may also include "unknowns" whose information was just not granular enough to be able to determine the histological subtype with absolute certainty
- c_histology_all: All observed histology recording for a given patient (optional if return_all = TRUE)

Examples

```
## Not run:
library(encore.io)

analysis_cohort_histology <- x |>
  edb2_get_histology(cancer = "NSCLC", histology_match = "adenocarcinoma")

## End(Not run)
```

---

edb2_get_labs *Query lab information for a given cohort*

Description

Function queries the lab table and standardizes according to a reference measurement unit.

Usage

```
edb2_get_labs(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC"),
  lab_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB2 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB2 data/files are located
cancer	character, one of MM or NSCLC
lab_name	character, curated name of lab (see details)
from	integer, left boundary of lab measurement window relative to index date (e.g., -90, indicating lab should be measured not before 90 days before index date)
to	integer, right boundary of lab measurement window relative to index date (e.g., 0, indicating lab should be measured not later than the day of the index date (inclusive))
ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" lab measurement be prioritized
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in labs_mapping_edb2
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function queries supported labs in the given from - to measurement window, selects the closest measurement to the index date, prioritizes one measurement in case of ties and standardizes the quantitative result to 1. a binary "normal" vs. "abnormal" variable (depending on if the measurement is inside or outside a given physiological reference range) and 2. standardizes the quantitative lab result according to a reference measurement unit (e.g., a result in g/dL is converted to a result in g/L, latter of which is the reference unit).

Note: Only selected labs are supported by this function which were taken from the ROPRO prognostic score (Becker T et al., Ann Oncol 2020).

The supported and available labs are:

- c_albumin_g_l (albumin mass/volume in serum or plasma)
- c_alp_u_l (alkaline phosphatase enzymatic activity/volume in serum or plasma)
- c_alt_u_l (alanine aminotransferase enzymatic activity/volume in serum or plasma)
- c_ast_u_l (aspartate aminotransferase enzymatic activity/volume in serum or plasma; used to compute ast-alt ratio)
- c_bilirubin_mg_dl (total bilirubin mass/volume in serum or plasma)
- c_calcium_mg_dl (calcium mass/volume in serum or plasma)
- c_hemoglobin_g_dl (Hemoglobin)
- c_ldh_u_l (lactate dehydrogenase enzymatic activity/volume in serum or plasma)
- c_neutrophil_10_9_l (neutrophils #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_platelets_10_9_l (platelets #/volume in blood)
- c_protein_g_l (protein Mass/volume in Serum or Plasma)

The following labs are part of ROPRO but are either not available in EDB2 or not supported yet:

- c_chloride_mmol_l (chloride `moles/volume` in serum or plasma)
- c_eosinophils_leukocytes_ratio (eosinophils/100 leukocytes in blood)
- c_glucose_mg_dl (Glucose)
- c_glucose_mg_dl (glucose `mass/volume` in serum or plasma)
- c_light_chain_kappa (Light Chain Kappa; dichotomous; >0 vs. 0)
- c_light_chain_lambda (Light Chain Lambda; dichotomous; >0 vs. 0)
- c_lymphocyte_10_9_l (lymphocytes `#/volume` in blood; used to compute neutrophil/lymphocyte ratio)
- c_lymphocyte_leukocyte_ratio (lymphocytes/100 leukocytes in blood)
- c_m_protein_igg (M protein IgG (dichotomous; >0 vs. 0)
- c_monocytes_10_9_l (monocytes `#/volume` in blood)
- c_urea_nitrogen_mg_dl (urea nitrogen `mass/volume` in serum or plasma)

The function operates via the following logic:

1. Pull lab table and subset to patients included in inception cohort `x`.
2. Subset to lab of interest.
3. The lab date is derived as the date of the lab test itself or, if missing, the documented date.
4. Lab test results are categorized into "normal" and "abnormal" depending on if they fall within the provided reference.
5. The quantitative results are standardized to the same reference unit using conversion factors which are mapped in the `labs_mapping_edb2` table.
6. Measurements with missing lab test result (or implicitly missing unit due to prior conversion step) are discarded.
7. if `set_implausible_na` is set TRUE:
    - a lab test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the `labs_mapping_implausible_values` table.
8. All lab dates are restricted to the ones that fall within the measurement window specified in the `from` and `to` arguments for the lower and upper limits of the measurement window, respectively.
9. If multiple lab tests are available for a given patient, the prioritization step is carried out as follows:
    - For each patient, lab tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if `ties` = "higher") or ascending quantitative results to prioritize the lower measurement (if `ties` = "lower") measurements. The default is "lower".
    - The first row for each patient after the above sorting is selected
10. The newly created variables are named according to the standardized lab name and unit of measurement and joined back to `x`.

Value

x with all additional labs variables joined, that is:

- `c_(lab_name)_distance_(from)_(to)` - days from date of lab measurements to index date
- `c_(lab_name)_(unit)_(from)_(to)` - binary lab result indicating if lab was within ("normal") the reference range or outside ("abnormal")
- `c_(lab_name)_(unit)_(from)_(to)_cont` - quantitative lab result after unit harmonization/conversion

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb2_get_labs(
    cancer = "NSCLC",
    lab_name = "c_albumin_g_l",
    from = -90,
    to = 0
    )

## End(Not run)
```

`edb2_get_os` *Query overall survival outcome for a given cohort*

Description

Function queries mortality and other information to derive a righ-censored time to all-cause mortality endpoint

Usage

```
edb2_get_os(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC"),
  data_cut_off_date = lubridate::ymd("2023-02-24"),
  verbose = TRUE
)
```

Arguments

x	dataframe queried from edb2 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where data/files are located. Default is path_edb2 environment variable if available.
cancer	character, one of MM or NSCLC
data_cut_off_date	date of database lock; according to vendor communication the data cut-off for the Q3 2023 delivery is Feb 24, 2023. This parameter can be changed if a grace period of x months before database lock is desired.
verbose	logical, print query progress and informative meta information

Details

The function queries and intention-to-treat (ITT) overall survival endpoint. The ITT follow-up time is defined as the time interval from index date (dt_index, including index date) to the date of death (if death event occured), or the date of a patient's last structured clinical activity or database cut-off, whichever is earlier. Upon advice by the EDB2 vendor, the documented or reported days were not used (except for ECOG) as this is not a good indicator if a patient was truly alive by that time or not. All tables with dates were used to derive the censoring date except for tumor grading as here only reported and documented dates were available.

For a fraction of patients in EDB2, only month-level or year-level granularity is provided for the dates used to compute follow-up. This can result in implausible/negative follow-up times if the index date is after the imputed date of death.

The function operates via the following logic:

1. Pull mortality details from the one-row-per-patient patient table and subset to patients included in inception cohort x.
2. A flag is created for patient with year-level granularity in the date of death.
3. The date of death is imputed by default to the mid of the month and the mid of the year for patient with month-level (default) and year-level granularity, respectively.
4. A patient's last structured activity date is determined based on all relevant tables (which cannot be listed for confidentiality and database de-identifications reasons).
5. Mortality and last structured activity dates are joined to x and a binary death event variable is derived based on if a date of death is captured for a given patient.
6. The censoring date is determined as the date of death, if available. If no death date is available, the censoring date is determined as the last structured activity (see step 4) or at the data cut-off date (23-02-24 is the default), whichever is earlier.
7. The follow-up (FU) time, defined as the time from index date (including index date) until censor date, is derived in days, months (= FU time in days/30.417) and years, depending on what the time scale should be for the respective analysis.
8. The newly created variables are joined back to x.

Value

x with all endpoint information joined, that is:

- death_itt, event indicator for all-cause mortality

- fu_itt_days, ITT follow-up time in days
- fu_itt_months, ITT follow-up time in months (i.e.,fu_itt_days / 30.417)
- fu_itt_years, ITT follow-up time in years

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb2_get_os(
    cancer = "NSCLC"
    )

## End(Not run)
```

---

edb2_get_vitals *Query vital sign measurements for a given cohort*

Description

The function queries vital sign measurements in a specified measurement window and converts standardized measurements to SI units (e.g., meters for height and kg for weight).

Usage

```
edb2_get_vitals(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "NSCLC"),
  vital_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE,
  verbose = FALSE
)
```

Arguments

x	dataframe queried from EDB2 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB2 data/files are located
cancer	character, one of MM or NSCLC
vital_name	character, curated name of vital sign measurement (see details)

`from` integer, left boundary of vital signs measurement window relative to index date (e.g., -90, indicating vital should be measured not before 90 days before index date)

`to` integer, right boundary of vital signs measurement window relative to index date (e.g., 0, indicating vital should be measured until day of index date (inclusive))

`ties` character, in case of ties (two equi-distant measurements), should the "higher" or "lower" vital test measurement be prioritized?

`set_implausible_na` logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in `vitals_mapping_edb2`

`label_name` logical, should variable name carry information about the measurement window

`verbose` logical, print query progress and informative meta information

The function operates via the following logic:

1. Pull vitals table and subset to patients included in inception cohort `x`.
2. Subset to vitals measurement of interest.
3. The vital test date is derived as the date of the lab test itself or, if missing, the documented date.
4. The quantitative height and weight are standardized to SI-units as documented in the `vitals_mapping_edb2` table.
5. Measurements with missing vital test result are discarded.
6. if `set_implausible_na` is set TRUE:
   - a vitals test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the `vitals_mapping_implausible_` table.
7. All vital dates are restricted to the ones that fall within the measurement window specified in the `from` and `to` arguments for the lower and upper limits of the measurement window, respectively.
8. If multiple vitals tests are available for a given patient, the prioritization step is carried out as follows:
   - For each patient, vitals tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if `ties` = "higher") or ascending quantitative results to prioritize the lower measurement (if `ties` = "lower") measurements. The default is "lower".
   - The first row for each patient after the above sorting is selected
9. The newly created variables are named according to the standardized vitals name and unit of measurement and joined back to `x`.

Details

The function queries and cleans all available vital sign measurements. In detail, the function selects measurements in a `from - to` measurement window relative to the index date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the `lower (default)` or `higher` measurement can be prioritized.

The available and supported vitals are:

- c_height (height in m)
- c_weight (weight in kg)

To compute BMI, query both c_height and c_weight and compute c_bmi as c_weight/c_height^2

Value

x with all additional vital test variables joined, that is:

- `c_(vital_name)_distance_(from)_(to)` - days from date of vital sign measurements to index date
- `c_(vital_name)_(unit)_(from)_(to)_cont` - quantitative vital sign measurement

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb2_get_vitals(
    cancer = "NSCLC",
    vital_name = "c_height",
    from = -180,
    to = 0
    )

## End(Not run)
```

---

edb2_path_helper *Helper function to assign correct paths based on cancer entity in EDB2*

Description

Use this helper function to assign the correct paths based on the cancer entity

Usage

```
edb2_path_helper(
  path = Sys.getenv("path_edb2"),
  cancer = c("MM", "MultipleMyeloma", "NSCLC")
)
```

Arguments

`path`	string, path to edb2 root directory
`cancer`	character, either "NSCLC" or "MM

Details

lot_path in MM also contains the imwg.csv table

Value

paths to line of therapy (lot_path) and all other data files (data_files_path)

Examples

```
## Not run:
library(encore.io)

nsclc_paths <- edb2_path_helper(cancer = "NSCLC")

## End(Not run)
```

`edb2_query_ropro` *Query and curate all relevant ROPRO variables*

Description

This function queries, cleans and curates all necessary covariates from EDB2 as they are used to compute the ROPRP prognostic score (Becker, Weberpals, et al., Ann Oncol 2020).

Usage

```
edb2_query_ropro(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb2"),
  cancer = NULL,
  from = -90,
  to = 0,
  verbose = TRUE
)
```

Arguments

`x`	dataframe queried from EDB2 with at least patient ids and therapy index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB2 data/files are located
`cancer`	character, one of MM or NSCLC
`from`	integer, left boundary of measurement window for time-dependent variables relative to index date (e.g., -90, indicating variables should be measured not before 90 days before index date)
`to`	integer, right boundary of measurement window for time-dependent variables relative to index date (e.g., 0, indicating variables should be measured not later than the day of the index date (inclusive))
`verbose`	logical, print progress of query

Details

Wrapper around major functions to query required covariates to compute ROPRO. Selected covariates are log transformed or log-log transformed. More details, see Becker, Weberpals, et al., Ann Oncol 2020.

Note that EDB2 does not provide all covariates to compute the full ROPRO model. Hence, this function queries all available covariates for a reduced model.

Value

x with all required ROPRO covariates joined. All general and cancer type-specific (as specified in `cancer` argument) covariates are returned.

Examples

```
## Not run:
library(encore.io)

x_ropro <- x |>
  edb2_query_ropro(
    index_date = "dt_index",
    path = Sys.getenv("path_edb2"),
    cancer = "NSCLC",
    from = -90,
    to = 0,
    verbose = TRUE
    )

## End(Not run)
```

---

edb3_get_biomarker *Query biomarker information for a given inception cohort*

Description

Function queries biomarker tables and curates information on alterations in defined driver genes.

Usage

```
edb3_get_biomarker(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  biomarker_name = NULL,
  from = -90,
  to = 0,
  label_name = FALSE
)
```

Arguments

`x`	dataframe queried from edb3 with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB3 data/files are located
`biomarker_name`	character, name of biomarker/gene alteration (see details)
`from`	integer, left boundary of biomarker measurement window relative to index date (e.g., -90, indicating biomarker should be measured not before 90 days before index date)

to	integer, right boundary of biomarker measurement window relative to index date (e.g., 0, indicating biomarker should be measured until day of index date (inclusive))
label_name	logical, should variable name carry information about the measurement window

Details

The function queries and categorizes a certain biomarker that was collected and curated as part of the database as either positive or negative. The categorization happens according to if the biomarker mutation/alteration is a clinically actionable one. For example, patients with any mutation in the EGFR gene or MSI-H/dMMR status would be classified as positive. In case there are multiple measurements per biomarker and patient, the time point of measurement is defined as the non-missing biomarker result in the covariate ascertainment window given by 'from' and 'to' that is closest to the index date.

The biomarker recorded date (`dt_(biomarker)`) in this database is defined as the date documented or result date. Note that the biomarker date is de-identified to week (date documented or result date) and is converted to ymd format.

Depending on the cancer type, the biomarker names can be: alk, braf, brca1, brca2, ddr2, egfr, esr1, fgfr1, kras mammaprint, met, mlh1, msh2, msh6, nras, ntrk1, ntrk2, ntrk3, pik3ca, pms2, ret, ros1, tp53

Value

x with all additional biomarker variables joined, that is:

- `c_(biomarker_name)_status_(from)_(to)` (binary, biomarker mutation status positive or negative)
- `c_(biomarker_name)_detail_(from)_(to)` (character string, more details about selected measurement)
- `c_(biomarker_name)_detail_all_(from)_(to)` (character string, this provides details about all results and details for this biomarker if there were multiple tests in the measurement window (from, to))
- `c_(biomarker_name)_distance_(from)_(to)` (numeric, relative distance between date of measurement and index date in days)
- `dt_(biomarker_name)_(from)_(to)` (date, date of selected biomarker measurement)

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)
analysis_cohort <- inception_cohort |>
  edb3_get_biomarker(
    cancer = "NSCLC",
    biomarker_name = "egfr",
    from = -180,
    to = 0)

## End(Not run)
```

`edb3_get_demographics` *Query demographic variables for an inception cohort in EDB3*

Description

Function queries all available demographic variables from the EDB3 database, curates them and joins them to the inception cohort x.

Usage

```
edb3_get_demographics(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3")
)
```

Arguments

`x`	dataframe queried from edb3 with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date
`path`	character string, path to directory where edb3 data/files are located

Details

The function queries and categorizes demographic variables as reported in the electronic health record (EHR). The function operates via the following logic:

Demographics:

1. Pull the initial diagnosis date of the primary cancer and subset to patients included in inception cohort `x`.
2. Only non-missing diagnosis dates that were chart-abstracted are considered.
3. If there are multiple diagnosis dates for a given patient, the earliest recorded date is used.
4. Pull the demographics table and join the diagnosis date and index date for each patient.
5. The patient date of birth is reported on year-level granularity and is imputed to the 1st July of the respective year (e.g. 1955 becomes 1955-07-01).
6. The age at initial diagnosis and at index date is computed as
   - The date of initial diagnosis - imputed date of birth
   - The index date - imputed date of birth
7. Additional categorical age variables are computed as <65 and 65+ years of age.
8. The reported sex, race and ethnicity variables are provided in a one-row-per-patient format and directly mapped one-to-one. If race is categorized as "Other or Unknown Race" or ethnicity is categorized to "Unknown", those values are set to missing.
9. Additionally, race and ethnicity are combined into a new combined variable `dem_race_ethnicity` with five mutually exclusive groups as defined by SEER.
10. The patient's practice type, major US geographic region and state are provided on a one-row-per-patient format and are propagated as recorded.

11. The newly created variables are joined back to x.

Smoking: Smoking history is derived based on LOINC code 72166-2. The smoking status is derived by filtering all LOINC codes available before or on index date. Smoking status is hierarchically mapped to being a current or former smoker (= smoking history), never smoker (= no smoking history) or being missing ('Other' are categorized as missing).

Value

x with all additional demographic variables joined (see details), that is,

- dem_age_initial_diagnosis (categorized age measured at initial cancer diagnosis: <60, 60-69, 70-79, 80+)
- dem_age_le_18_flag (logical indicating if patients was at least (larger/equal; le) 18 years of age at index date)
- dem_age_index_cont (continuous age measured at index date; derived from from dt_index and dt_dob_imputed)
- dem_age_index (categorized age measured at index date: <60, 60-69, 70-79, 80+)
- dem_sex (binary, Male, Female, NA)
- dem_race (categorical, "", "Declined" and "Other" are converted to NA)
- dem_ethnicity (binary, "" and "Declined" are converted to NA)
- dem_race_ethnicity (categorical, classification into five mutually exclusive groups according to SEER)
- dem_practice (academic/community hospital patients receives care)
- dem_region (categorical, Northeast, South, West, Midwest, Multiple)
- dem_state (categorical, state patients receives care)
- c_smoking_history (binary, history of smoking on or before index date, 1 = current or former, 0 = never)

Examples

```
## Not run:
library(encore.io)
analysis_cohort <- inception_cohort |>
  edb3_get_demographics(cancer = "NSCLC")

## End(Not run)
```

---

edb3_get_diagnosis_solid

Query initial and metastatic diagnosis dates for EDB3 database

---

Description

Function queries diagnosis details including staging information.

Usage

```
edb3_get_diagnosis_solid(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3")
)
```

Arguments

x	dataframe queried from edb3 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB3 data/files are located

Details

Function queries diagnosis details for the index cancer in EDB1 for all patients included in x. Disease-specific ICD-10-CM codes are used to screen patients for review. All patients are subsequently confirmed to have the appropriate diagnosis during the curation process using staging, histology, and other information available within the patient documents. Additionally, all patients are confirmed to be >=18 years old on the date of the initial diagnosis. To determine the initial cancer diagnosis, pathologic staging is always prioritized where available.

The function operates via the following logic:

1. Pull diagnosis and disease details from disease-specific table and subset to patients included in inception cohort x.
2. Records are filtered for cases with an ICD-10-CM code starting with C50 and non-missing diagnosis date.
3. Select all diagnoses that are curated and the row that has the earliest diagnosis date. If there are duplicate recordings, they are removed.
4. Only patients are retained that are in (inner join).
5. The stage is derived as the ...

Value

x with all additional diagnosis variables joined, that is:

- dt_initial_dx (date, date of diagnosis or first documented diagnosis date for tumor)
- c_stage_initial_dx (nominal, summary group stage at initial diagnosis, if available)
- dt_met_dx (date, date of earliest evidence of distant metastasis)
- c_de_novo_mets_dx (binary logical, evidence of presence of one or multiple metastases at/before initial diagnosis)
- c_time_dx_to_index (continuous, time between initial diagnosis and index date (in days))
- c_time_met_dx_to_index (continuous, time between earliest evidence of a metastatic diagnosis and index date (in days)
- c_met_pre_index (binary logical, evidence of any metastasis before/on index date; includes de novo metastatic patients and progressors (overlap with c_de_novo_mets_dx possible))
- c_number_met_sites (integer, number of metastatic sites for a given patient anytime before/on index date (inferred from ICD codes, see icd_metastases system file)
- c_met_sites (character string, description of anatomical locations of metastatic sites for a given patient patient's index date)

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb3_get_diagnosis_solid(
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  cancer = "MetastaticBreast"
  )


## End(Not run)
```

---

edb3_get_ecog *Query performance status information for an inception cohort*

Description

Function queries performance status (ECOG, Karnofsky) tables and curates derived variables

Usage

```
edb3_get_ecog(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  from = -90,
  to = 0,
  ties = "lower",
  label_name = FALSE
)
```

Arguments

x	dataframe queried from edb3 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB3 data/files are located
from	integer, left boundary of ECOG measurement window relative to index date (e.g., -90, indicating ECOG should be measured not before 90 days before index date)
to	integer, right boundary of ECOG measurement window relative to index date (e.g., 0, indicating ECOG should be measured until day of index date (inclusive))
ties	character, one of "lower" or "higher" to choose either the lower (default) or higher ECOG measurement if there are two measurements on the same day
label_name	logical, should variable name carry information about the measurement window

Details

The function queries all ECOG and Karnofsky measurements, then maps the Karnofsky measurment to an ECOG value according to Oken et al. (Am J Clin Oncol 1982), then filters for all measurements within the indicated time window specified by `from` and `to`. It then chooses the measurement closest to the index date (closest relative distance). In case of ties, the user has the option to choose the lower or higher (`ties`) measurement.

Value

x with all additional ECOG variables joined, that is:

- `c_ecog_{from}_{to}`, ECOG value measured in the specified measurement window
- `c_ecog_{from}_{to}`, distance of the date the ECOG value was measured relative to the index date

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb3_get_ecog(
    cancer = "NSCLC",
    from = -180,
    to = 0,
    ties = "lower"
    )

## End(Not run)
```

---

edb3_get_histology *Query histology information from edb3*

Description

Function queries and binarizes information for a provided histologiucal subtype from the EDB3 database. Note: This function does not apply to multiple myeloma

Usage

```
edb3_get_histology(
  x = NULL,
  path = Sys.getenv("path_edb3"),
  histology_match = NULL,
  return_all = FALSE
)
```

Arguments

x	dataframe queried from edb3 with at least patient ids and index date of inception cohort
path	character string, path to directory where EDB3 data/files are located
histology_match	character, string match to categorizeand identify patients with a certain histology, e.g. "adenocarcinoma"
return_all	logical, should a variable be returned that summarizes all recorded histology measurements? default is FALSE

Details

column information includes: Many patients can have more than one histological subtype recorded. In this function, the user must provide the desired cancer type and histological subtype and returns a binary TRUE/FALSE if any of the histological recordings match the histology subtype provided in histology_match as well as a summary of all recorded histological subtypes (optional). Tip: the function can also be stacked/executed multiple times with different histological subtypes.

CAVE: EDB3 does not provide any information about the specimen/tissue type for which the histology was determined!

Note that the search string defined in argument `histology_match` is not case sensitive.

Value

x with all additional histology variables joined, prepended with "c_"

- `c_(histology_match)`: A TRUE/FALSE if the histological subtype was observed for a given patient. FALSE may also include "unknowns" whose information was just not granular enough to be able to determine the histological subtype with absolute certainty
- `c_histology_all`: All observed histology recording for a given patient (optional if return_all = TRUE)

Examples

```
## Not run:
library(encore.io)
analysis_cohort_histology <- x |>
  edb3_get_histology(histology_match = "adenocarcinoma")

## End(Not run)
```

---

edb3_get_labs *Query lab information for a given cohort*

Description

Function queries the lab table and standardizes according to a reference measurement unit.

Usage

```
edb3_get_labs(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  lab_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB3 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB3 data/files are located
lab_name	character, curated name of lab (see details)
from	integer, left boundary of lab measurement window relative to index date (e.g., -90, indicating lab should be measured not before 90 days before index date)
to	integer, right boundary of lab measurement window relative to index date (e.g., 0, indicating lab should be measured not later than the day of the index date (inclusive))
ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" lab measurement be prioritized
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in `labs_mapping_edb3`
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function queries and cleans supported lab measurements. In detail, the function selects measurements in a `from - to` measurement window relative to the index date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the `lower (default)` or `higher` measurement can be prioritized.

Note: Only selected labs are supported by this function which were taken from the ROPRO prognostic score (Becker T et al., Ann Oncol 2020).

The supported and available labs are:

- c_albumin_g_l (albumin `mass/volume` in serum or plasma)
- c_alp_u_l (alkaline phosphatase `enzymatic activity/volume` in serum or plasma)
- c_alt_u_l (alanine aminotransferase `enzymatic activity/volume` in serum or plasma)

- c_ast_u_l (aspartate aminotransferase enzymatic activity/volume in serum or plasma; used to compute ast-alt ratio)
- c_bilirubin_mg_dl (total bilirubin mass/volume in serum or plasma)
- c_calcium_mg_dl (calcium mass/volume in serum or plasma)
- c_chloride_mmol_l (chloride moles/volume in serum or plasma)
- c_eosinophils_leukocytes_ratio (eosinophils/100 leukocytes in blood)
- c_glucose_mg_dl (glucose mass/volume in serum or plasma)
- c_granulocytes_leukocytes_ratio (granulocytes/100 leukocytes in blood)
- c_hemoglobin_g_dl (hemoglobin mass/volume in blood)
- c_ldh_u_l (lactate dehydrogenase enzymatic activity/volume in serum or plasma)
- c_lymphocyte_10_9_l (lymphocytes #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_lymphocyte_leukocyte_ratio (lymphocytes/100 leukocytes in blood)
- c_monocytes_10_9_l (monocytes #/volume in blood)
- c_neutrophil_10_9_l (neutrophils #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_platelets_10_9_l (platelets #/volume in blood)
- c_protein_g_l (protein mass/volume in Serum or Plasma)
- c_urea_nitrogen_mg_dl (urea nitrogen mass/volume in serum or plasma)

The function operates via the following logic:

1. Pull lab table and subset to patients included in inception cohort x.
2. Subset to lab of interest.
3. The lab date is derived as the date of the lab test itself.
4. Lab test results are categorized into "normal" and "abnormal" depending on if they fall within the provided reference.
5. The quantitative results are standardized to the same reference unit using conversion factors which are mapped in the labs_mapping_edb3 table.
6. Measurements with missing lab test result (or implicitly missing unit due to prior conversion step) are discarded.
7. if set_implausible_na is set TRUE:
    - a lab test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the labs_mapping_implausible_values table.
8. All lab dates are restricted to the ones that fall within the measurement window specified in the from and to arguments for the lower and upper limits of the measurement window, respectively.
9. If multiple lab tests are available for a given patient, the prioritization step is carried out as follows:
    - For each patient, lab tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if ties = "higher") or ascending quantitative results to prioritize the lower measurement (if ties = "lower") measurements. The default is "lower".
    - The first row for each patient after the above sorting is selected
10. The newly created variables are named according to the standardized lab name and unit of measurement and joined back to x.

Value

x with all additional labs variables joined, that is:

- `c_(lab_name)_distance_(from)_(to)` - days from date of lab measurements to index date
- `c_(lab_name)_(unit)_(from)_(to)` - quantitative lab result after unit harmonization/conversion

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb3_get_labs(
    lab_name = "c_albumin_g_l",
    from = -90,
    to = 0
    )

## End(Not run)
```

`edb3_get_os` *Query overall survival outcome for a given inception cohort*

Description

Function queries mortality and other information to derive a righ-censored time to all-cause mortality endpoint

Usage

```
edb3_get_os(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  data_cut_off_date = lubridate::ymd("2024-09-30"),
  verbose = TRUE
)
```

Arguments

`x`	dataframe queried from edb3 with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where data/files are located
`data_cut_off_date`	date of database lock; the data cut-off for this delivery is Sep 30, 2024. This parameter can be changed if a grace period of x months before database lock is desired.
`verbose`	logical, print query progress and informative meta information

Details

The function queries and curates intention-to-treat (ITT) overall survival endpoint. The ITT follow-up time is defined as the time interval from index date (`dt_index`) to the date of death (if death event occured), or the last date of proof that the patient was alive at that time (date de-identified to week).

Value

x with all endpoint information joined, that is:

- `death_itt`, event indicator for all-cause mortality
- `fu_itt_days`, ITT follow-up time in days
- `fu_itt_months`, ITT follow-up time in months (i.e.,`fu_itt_days` / 30.417)
- `fu_itt_years`, ITT follow-up time in years

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb3_get_os()

## End(Not run)
```

`edb3_get_vitals` *Query vital sign measurements for a given inception cohort*

Description

Function queries vitals sign measurements

Usage

```
edb3_get_vitals(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  vital_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE
)
```

Arguments

x	dataframe queried from EDB3 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB3 data/files are located
vital_name	character, curated name of vital sign measurement (see details)
from	integer, left boundary of vital test measurement window relative to index date (e.g., -90, indicating vital test should be measured not before 90 days before index date)
to	integer, right boundary of vital test measurement window relative to index date (e.g., 0, indicating vital test should be measured not later than the day of the index date (inclusive))
ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" vital test measurement be prioritized?
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in vitals_mapping_edb3
label_name	logical, should variable name carry information about the measurement window

Details

The function queries and cleans supported vital sign measurements. In detail, the function selects measurements in a from - to measurement window relative to the index date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the lower (default) or higher measurement can be prioritized.

The supported vitals are:

- c_sbp (systolic blood pressure in mmHg)
- c_dbp (diastolic blood pressure in mmHg)
- c_bmi (body mass index in kg/m^2 directly measured or derived from weight/height^2)
- c_height (height in m)
- c_hr (heart rate/pulse in beats/min)
- c_oxygen (oxygen saturation; taken from O2 sat and pulse oximetry)
- c_resp (respiration in breaths/min)
- c_weight (weight in kg)

The function operates via the following logic:

1. Pull vitals table and subset to patients included in inception cohort x.
2. Subset to vitals measurement of interest.
3. The quantitative results are standardized to the same reference unit using conversion factors which are mapped in the vitals_mapping_edb3 table.
4. Measurements with missing vital test result are discarded.
5. if set_implausible_na is set TRUE:
    - a vitals test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the vitals_mapping_implausible_values table.

6. The vitals test date is derived as the date of the vitals test itself.
7. All vital dates are restricted to the ones that fall within the measurement window specified in the `from` and `to` arguments for the lower and upper limits of the measurement window, respectively.
8. If multiple vitals tests are available for a given patient, the prioritization step is carried out as follows:
   - For each patient, vitals tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if `ties` = "higher") or ascending quantitative results to prioritize the lower measurement (if `ties` = "lower") measurements. The default is "lower".
   - The first row for each patient after the above sorting is selected
9. The newly created variables are named according to the standardized vital test name and unit of measurement and joined back to `x`.

Value

x with all additional vital test variables joined, that is:

- `c_(vital_name)_distance_(from)_(to)` - days from date of vital sign measurements to index date
- `c_(vital_name)_(unit)_(from)_(to)_cont` - quantitative vital sign measurement

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb3_get_vitals(
    vital_name = "c_weight",
    from = -90,
    to = 0
    )

## End(Not run)
```

`edb3_query_ropro` *Query and curate all available ROPRO variables*

Description

This function queries, cleans and transforms all necessary covariates needed to compute the ROPRO prognostic score in EDB3.

Usage

```
edb3_query_ropro(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb3"),
  from = -90,
  to = 0,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB3 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index date, default is dt_index
path	character string, path to directory where EDB3 data/files are located
from	integer, left boundary of measurement window for time-dependent variables relative to index date (e.g., -90, indicating variables should be measured not before 90 days before index date)
to	integer, right boundary of measurement window for time-dependent variables relative to index date (e.g., 0, indicating variables should be measured not later than the day of the index date (inclusive))
verbose	logical, print progress of query

Details

Wrapper around major functions to query required covariates to compute ROPRO. Selected covariates are log transformed or log-log transformed. More details, see Becker, Weberpals, et al., Ann Oncol 2020.

Value

x with all required ROPRO covariates joined. All general and cancer type-specific (as specified in `cancer` argument) covariates are returned.

Note that:

- covariates for BC are identical to the the general pan-tumor ROPRO, just the weights are different

Examples

```
## Not run:
library(encore.io)

x_ropro <- x |>
  edb3_query_ropro(
    index_date = "dt_index",
    path = Sys.getenv("path_edb3"),
    from = -90,
    to -0,
    verbose = TRUE
    )
```

```
## End(Not run)
```

---

edb4_get_biomarker *Query biomarker information for a given inception cohort*

Description

Function queries biomarker tables and curates information on alterations in defined driver genes.

Usage

```
edb4_get_biomarker(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "MM", "NSCLC"),
  biomarker_name = NULL,
  from = -90,
  to = 0,
  label_name = FALSE
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC, MM or NSCLC
biomarker_name	character, name of biomarker/gene alteration (see details)
from	integer, left boundary of biomarker measurement window relative to index date (e.g., -90, indicating biomarker should be measured not before 90 days before index date)
to	integer, right boundary of biomarker measurement window relative to index date (e.g., 0, indicating biomarker should be measured until day of index date (inclusive))
label_name	logical, should variable name carry information about the measurement window

Details

The function queries and categorizes a certain biomarker that was collected and curated as part of the database as either positive or negative. The categorization happens according to if the biomarker mutation/alteration is a clinically actionable one. For example, patients with any mutation in the EGFR gene or MSI-H/dMMR status would be classified as positive. In case there are multiple measurements per biomarker and patient, the time point of measurement is defined as the non-missing biomarker result in the covariate ascertainment window given by `from` and `to` that is closest

to the index_date. The biomarker recorded date (dt_(biomarker)) in this database is defined as the date documented or result date. Note that the biomarker date is de-identified to week (date documented or result date) and is converted to ymd format.

Depending on the cancer type, the biomarker names can be: alk, braf, brca1, brca2, egfr, er, esr1, her2neu, kras, met, mmr, msi, nras, ntrk1, ntrk2, ntrk3, pd-l1, pik3ca, pr, ret, ros1, tmb

The function operates via the following logic:

1. Pull biomarker table and subset to patients included in inception cohort x.
2. Subset table to biomarker names as specified in biomarker_name.
3. Biomarker status test result is categorized hierarchically:
   - If biomarker value matches any of "wild type" OR "negative" OR "proficient" OR "non-high" OR "fusion not detected" OR "no", the result is mapped to "negative".
   - If biomarker value matches any of "mutation" OR "mutant" OR "2+" OR "positive" OR "deficient" OR "high" OR "fusion detected" OR "yes", the result is mapped to "positive".
   - If none of the above applies, the result is mapped to NA (missing).
4. Missing mapped biomarker status results are discarded.
5. The biomarker date is specified as the result or documented date in ymd format (the two cannot be further distinguished as per data documentation and is de-identified to week).
6. Biomarker dates outside of the specified measurement window relative to the index date (defined using the index_date, from and to arguments) are discarded.
7. If multiple biomarker tests are available for a given patient, the prioritization step is carried out as follows:
   - For each patient, biomarker mappings (negative, positive) and the absolute distance from biomarker date to index date are sorted in descending and ascending order, respectively.
   - To prioritize any positive biomarker mapping closest to the index date, the first row for each patient is selected. This reflects the final mapped biomarker status variable.
   - In addition, all available biomarker details within the measurement window are collapsed into a new variable. This reflects the final biomarker detail variable.
8. The newly created biomarker variables are named accordingly and returned.
9. The newly created variables are joined back to x.

WARNING: the categorization of more complex biomarkers like HER2 requires a more refined approach or de novo written code to account for equivocal results.

Value

x with all additional biomarker variables joined, that is:

- c_(biomarker_name)_status_(from)_(to) (binary, biomarker mutation status positive or negative)
- c_(biomarker_name)_detail_(from)_(to) (character string, more details about selected measurement)
- c_(biomarker_name)_detail_all_(from)_(to) (character string, this provides details about all results and details for this biomarker if there were multiple tests in the measurement window (from, to))
- c_(biomarker_name)_distance_(from)_(to) (numeric, relative distance between date of measurement and index date in days)
- dt_(biomarker_name)_(from)_(to) (date, date of selected biomarker measurement)

Note: if from or to is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- inception_cohort |>
  edb4_get_biomarker(
    cancer = "NSCLC",
    biomarker_name = "egfr",
    from = -180,
    to = 0)

## End(Not run)
```

`edb4_get_demographics` *Query demographic variables for an inception cohort*

Description

Demographics:

1. Pull the initial diagnosis date of the primary cancer and subset to patients included in inception cohort `x`.
2. Pull the demographics table and join the diagnosis date and index date for each patient.
3. The patient date of birth is reported on year-level granularity and is imputed to the 2nd July of the respective year (e.g. 1955 becomes 1955-07-02).
4. The age at initial diagnosis and at index date is computed as
   - The date of initial diagnosis - imputed date of birth
   - The index date - imputed date of birth
5. Additional categorical age variables are computed as <65 and 65+ years of age.
6. The reported sex, family history (mapped to a logical TRUE/FALSE), geographic region, race and ethnicity variables are provided in a one-row-per-patient format and directly mapped one-to-one. If race is categorized as "Declined" or "Other" or ethnicity is categorized to "Declined", those values are set to missing.
7. Additionally, race and ethnicity are combined into a new combined variable `dem_race_ethnicity` with five mutually exclusive groups as defined by SEER. As opposed to other datasets, EDB1 has a larger "Other" race category since "American Indian or Alaska Native" seems to be not coded as such, i.e, "Other" is an explicit group in this case.
8. The newly created variables are joined back to `x`.

Usage

```
edb4_get_demographics(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "MM", "NSCLC")
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC, MM or NSCLC

Details

Smoking: The recorded smoking status is filtered to available records before or on index date. Smoking status is hierarchically mapped to being a current or former smoker (= smoking history), never smoker (= no smoking history) or being missing.

Value

x with all additional demographic variables joined (see details), that is,

- dem_age_initial_diagnosis (nominal, categorized age measured at initial cancer diagnosis: <60, 60-69, 70-79, 80+)
- dem_age_le_18_flag (logical, indicating if patients was at least (larger/equal; le) 18 years of age at index date)
- dem_age_index_cont (continuous, age measured at index date; note: date of bith has only year-granularity, hence age is imprecise)
- dem_age_index (nominal, categorized age measured at index date: <60, 60-69, 70-79, 80+)
- dem_sex (binary, Male, Female)
- dem_family_history (binary, TRUE, FALSE)
- dem_race (nominal, "", "Declined" and "Other" are converted to NA)
- dem_ethnicity (binary, "" and "Declined" are converted to NA)
- dem_race_ethnicity (nominal, classification into five mutually exclusive groups according to SEER)
- dem_region (nominal, region of the center/network the patient is receiving care at, can be Midwest, Northeast, South, West)
- c_smoking_history (logical, history of smoking on or before index date, TRUE = current or former, FALSE = never)

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb4_get_demographics(
    index_date = "dt_index",
    cancer = "NSCLC")

## End(Not run)
```

`edb4_get_diagnosis_heme` *Query diagnostic details for heme tumors in EDB4 database*

Description

Function queries diagnosis details including ISS staging information.

Usage

```
edb4_get_diagnosis_heme(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = "MM"
)
```

Arguments

`x`	dataframe queried from edb4 with at least patient ids and index date of inception cohort
`index_date`	character, variable/column name with the patient's index_date, default is dt_index
`path`	character string, path to directory where EDB4 data/files are located
`cancer`	character, so "MultipleMyeloma"

Details

Function queries diagnosis details for the index cancer in EDB4 for all patients included in `x`. Patients require an office visit and are sampled if they were diagnosed with one of the eligible index cancers and had a documented visit date within the reporting period (10/01/2018 through 09/30/2023) to one of the network facilities and were at least 20 years of age at the time of first diagnosis. Patients who were on a clinical trial at any point in their treatment history are excluded.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort `x`.
2. For each patient the dates of initial cancer diagnosis and staging are provided and converted to YMD format.
3. If an (ISS) group stage is recorded for MM, this information returned as `c_stage_initial_dx` as provided in the table. If the value equals any of "Other", "Unknown", a missing value is returned.
4. The time from initial cancer diagnosis to the index date is computed as `index_date - dt_initial_dx`.
5. Experimental: The number and location of potential secondary malignancies is inferred by C77.x, C78.x and C79.x ICD-10 codes (and corresponding ICD-9 mappings) from EDB4's ICD-9/ICD-10 diagnosis table. The `c_number_met_sites` is inferred by counting the unique number of sites as given by a C77.x, C78.x, C79.x granularity in the secondary diagnosis table. For more see `icd_metastases` system file. The resulting `c_number_met_sites` and `c_met_sites` are highly experimental variables and should be used with caution.

6. Identification of a specific multiple myeloma subgroup ('kappa light chain', 'lambda light chain', 'M protein IgG')
   - the labs table is used to identify lab tests related to 'kappa light chain', 'lambda light chain', and '^igg'
   - for each patient, membership to a mutually exclusive marker/subgroup is determined
   - the closes measurement before or on index date is prioritized
7. The newly created variables are joined back to `x`.

Value

x with all additional diagnosis variables joined, that is:

- `dt_initial_dx` (date, date of diagnosis or first documented diagnosis date for tumor)
- `c_stage_initial_dx` (nominal, summary ISS stage at initial diagnosis)
- `c_time_dx_to_index` (continuous, time between initial diagnosis and index date (in days))
- `c_m_protein_igg` (logical, whether the patient's immunoglobulin class of M protein is IgG)
- `c_light_chain_kappa` (logical, whether the patient's involved light chain is Kappa)
- `c_light_chain_lambda` (logical, Whether the patient's involved light chain is Lambda)

Experimental:

- `c_number_met_sites` (integer, number of metastatic sites for a given patient anytime before/on index date (inferred from ICD codes, see `icd_metastases` system file)
- `c_met_sites` (character string, description of anatomical locations of metastatic sites for a given patient patient's index date values are separated by ':,')

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb4_get_diagnosis_heme(
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = "MultipleMyeloma"
  )


## End(Not run)
```

---

edb4_get_diagnosis_solid

Query initial and metastatic diagnosis dates for EDB4 database

---

Description

Function queries diagnosis details including staging information.

Usage

```
edb4_get_diagnosis_solid(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "NSCLC")
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC, MM or NSCLC

Details

Function queries diagnosis details for the index cancer in EDB4 for all patients included in x. Patients require an office visit and are sampled if they were diagnosed with one of the eligible index cancers and had a documented visit date within the reporting period (10/01/2018 through 09/30/2023) to one of the network facilities and were at least 20 years of age at the time of first diagnosis. Patients who were on a clinical trial at any point in their treatment history are excluded.

The function operates via the following logic:

1. Pull diagnosis and disease details from the one-row-per-patient disease-specific table and subset to patients included in inception cohort x.
2. For each patient the dates of initial cancer diagnosis, staging and metastatic diagnosis (earliest evidence of metastasis) are provided and converted to YMD format.
3. If a summary group stage is recorded for a given cancer, this information returned as c_stage_initial_dx as provided in the table. If the value equals any of "Other", "Unknown", a missing value is returned.
4. The time from initial cancer diagnosis to the index date is computed as index_date - dt_initial_dx.
5. A boolean de novo metastatic status (c_de_novo_mets_dx) is derived hierarchically and is returned as
   - TRUE if a patient is diagnosed in stage IV, the TNM stage contains M1 or if the metastatic diagnosis date is on or precedes the initial diagnosis date
   - a missing value (NA) if both the stage at initial diagnosis, TNM stage and the date of metastatic diagnosis are missing
   - FALSE if none of the above apply
6. The time from initial cancer diagnosis to the index date is computed as index_date - dt_initial_dx.
7. The time from metastatic cancer diagnosis to the index date is computed as index_date - dt_met_dx (not applicable to patients without a metastasis).
8. Metastatic sites are provided by in the table as a string separated by '+' symbols and the number of metastatic sites per patient are derived by counting the amount of sites between the '+' delimiter (not applicable to patients without a metastasis).
9. A boolean c_met_pre_index variable is derived which indicates if there is evidence of a metastasis at anytime before or on the index date (which includes de novo metastatic patients AND progressors if applicable). This is derived hierarchically and is returned as

- TRUE if the metastatic diagnosis date coincidences or precedes the index date or if the patient has a de novo metastatic status
- a missing value (NA) if both the summary group stage at initial diagnosis and the TNM stage are missing
- FALSE if none of the above apply

10. The newly created variables are joined back to `x`.

Value

x with all additional diagnosis variables joined, that is:

- `dt_initial_dx` - Date of diagnosis or first documented diagnosis date for tumor (de-identified to week)
- `dt_staging` - Date stage was recorded, if available (de-identified to week)
- `dt_met_dx` - Date of earliest evidence of distant metastasis (de-identified to week)
- `c_stage_initial_dx` - First summary group stage at initial diagnosis, if available
- `c_tnm_initial_dx` - Individual TNM staging values/stages at initial diagnosis
- `c_de_novo_mets_dx` - Evidence of presence of one or multiple metastases at/before initial diagnosis
- `c_time_dx_to_index` - Time between initial diagnosis and index date (in days)
- `c_time_met_dx_to_index` - Time between earliest evidence of a metastatic diagnosis and index date (in days)
- `c_met_pre_index` - Evidence of any metastasis before/on index date (logical); includes de novo metastatic patients and progressors (overlap with c_de_novo_mets_dx possible)
- `c_number_met_sites` - number of metastatic sites for a given patient anytime before/on index date (inferred from `mets_location` as provided by the vendor)
- `c_met_sites` - description of anatomical locations of metastatic sites for a given patient patient's `c_number_met_sites`

Examples

```
## Not run:
library(encore.io)
analysis_cohort <- x |>
 edb4_get_diagnosis(cancer = "NSCLC")


## End(Not run)
```

`edb4_get_ecog` *Query performance status information for an inception cohort*

Description

Function queries performance status (ECOG, Karnofsky) tables and curates derived variables

Usage

```
edb4_get_ecog(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "MM", "NSCLC"),
  from = -90,
  to = 0,
  ties = "lower",
  label_name = FALSE
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC, MM or NSCLC
from	integer, left boundary of ECOG measurement window relative to index date (e.g., -90, indicating ECOG should be measured not before 90 days before index date)
to	integer, right boundary of ECOG measurement window relative to index date (e.g., 0, indicating ECOG should be measured until day of index date (inclusive))
ties	character, one of "lower" or "higher" to choose either the lower (default) or higher ECOG measurement if there are two measurements on the same day
label_name	logical, should variable name carry information about the measurement window

Details

The function queries all ECOG and Karnofsky measurements, then maps the Karnofsky measurment to an ECOG value according to Oken et al. (Am J Clin Oncol 1982), then filters for all measurements within the indicated time window specified by `from` and `to`. It then chooses the measurement closest to the index date (closest relative distance). In case of ties, the user has the option to choose the lower or higher (`ties`) measurement.

The function operates via the following logic:

1. Pull performance score measurements from structured sources and subset to patients included in inception cohort `x`.
2. Define the performance score date as the earliest of the performance score reported or documented timedelta relative to the `index_date`.
3. In rare cases, there are performances scores >5 that are erroneously labelled as ECOG and performance scores with values between 1-5 that are labelled as Karnofsky scores; these labels get cleaned to the respective source of the performance score.
4. Karnofsky performance scores are mapped to their ECOG score equivalents as per Oken et al. (Am J Clin Oncol 1982).
5. Subset to ECOG measurements within the specified measurement window (via `from` and `to`) relative to `index_date` and discard measurements with missing ECOG value.

6. Prioritize ECOG measurements assessed in closest absolute distance to the `index_date`.
   - if `ties = lower`: for each patient, arrange ECOG measurement by ascending absolute distance to index and ascending ECOG value (= lowest value) and pick the first observation
   - if `ties = higher`: for each patient, arrange ECOG measurement by ascending absolute distance to index and descending ECOG value (= highest value) and pick the first observation
7. The newly created variables are joined back to `x`.

Value

x with all additional ECOG variables joined, that is:

- `c_ecog_{from}_{to}`, ECOG value measured in the specified measurement window
- `c_ecog_{from}_{to}`, distance of the date the ECOG value was measured relative to the index date

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

analysis_cohort <- x |>
  edb4_get_ecog(
    cancer = "NSCLC",
    from = -180,
    to = 0,
    ties = "lower"
    )

## End(Not run)
```

`edb4_get_histology` *Query histology information for an inception cohort*

Description

Function queries histology information for a solid tumor inception cohort.

Usage

```
edb4_get_histology(
  x = NULL,
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "NSCLC"),
  histology_match = NULL,
  return_all = FALSE
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC or NSCLC
histology_match	character, string match to categorizeand identify patients with a certain histology, e.g. "adenocarcinoma"
return_all	logical, should a variable be returned that summarizes all recorded histology measurements? default is FALSE

Details

The function queries histology measurements if available for a given cancer. In EDB4, many patients can have more than one histological subtype recorded without indication of a specific date. The function prioritizes histology measurements that match the keyword search string in `histology_match`.

The function operates via the following logic:

1. Pull histology recording details from the histology table and subset to patients included in inception cohort `x`.
2. Create a binary boolean variable (TRUE/FALSE) indicating if the histology description/text string matches the keyword string in `histology_match`, e.g., a patient with a "squamous cell carcinoma" histology would return TRUE if "squamous" was the `histology_match` keyword string. The keyword search string defined in argument `histology_match` is not case sensitive and missing measurements are discarded. The name of the variable is adapted to match the keyword string.
3. To receive one measurement per patient, histology measurements are prioritized that match the keyword search string in `histology_match` (i.e., return TRUE). In case of ties, measurements with non-missing grading values are prioritized.
4. All available histology measurement for a given patient are summarized and returned in `c_histology_all` if `return_all` is specified as TRUE.
5. The newly created variables are joined back to `x`.

Tip: the function can also be stacked/executed multiple times with different histological subtypes.

column information includes: Many patients can have more than one histological subtype recorded. In this function, the user must provide the desired cancer type and histological subtype and returns a binary TRUE/FALSE if any of the histological recordings match the histology subtype provided in histology_match as well as a summary of all recorded histological subtypes (optional). Tip: the function can also be stacked/executed multiple times with different histological subtypes.

CAVE: EDB4 does not provide any information about the specimen/tissue type for which the histology was determined!

Note that the search string defined in argument `histology_match` is not case sensitive.

Value

x with all additional histology variables joined, prepended with "c_"

- `c_(histology_match)`: A TRUE/FALSE if the histological subtype was observed for a given patient. FALSE may also include "unknowns" whose information was just not granular enough to be able to determine the histological subtype with absolute certainty

- c_histology_all: All observed histology recording for a given patient (optional if return_all = TRUE)

Examples

```
## Not run:
library(encore.io)
analysis_cohort_histology <- x |>
  edb4_get_histology(cancer = "NSCLC", histology_match = "adenocarcinoma")

## End(Not run)
```

`edb4_get_labs` *Query lab information for a given inception cohort*

Description

Function queries the lab table and standardizes according to a reference measurement unit.

Usage

```
edb4_get_labs(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "MM", "NSCLC"),
  lab_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB4 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC, MM or NSCLC
lab_name	character, curated name of lab (see details)
from	integer, left boundary of lab measurement window relative to index date (e.g., -90, indicating lab should be measured not before 90 days before index date)
to	integer, right boundary of lab measurement window relative to index date (e.g., 0, indicating lab should be measured not later than the day of the index date (inclusive))

ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" lab measurement be prioritized
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in labs_mapping_edb4
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function queries and cleans supported lab measurements. In detail, the function selects measurements in a from - to measurement window relative to the index date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the lower (default) or higher measurement can be prioritized.

Note: Only selected labs are supported by this function which were taken from the ROPRO prognostic score (Becker T et al., Ann Oncol 2020).

The supported and available labs are:

- c_albumin_g_l (albumin mass/volume in serum or plasma)
- c_alp_u_l (alkaline phosphatase enzymatic activity/volume in serum or plasma)
- c_alt_u_l (alanine aminotransferase enzymatic activity/volume in serum or plasma)
- c_ast_u_l (aspartate aminotransferase enzymatic activity/volume in serum or plasma; used to compute ast-alt ratio)
- c_bilirubin_mg_dl (total bilirubin mass/volume in serum or plasma)
- c_calcium_mg_dl (calcium mass/volume in serum or plasma)
- c_chloride_mmol_l (chloride moles/volume in serum or plasma)
- c_eosinophils_leukocytes_ratio (eosinophils/100 leukocytes in blood)
- c_glucose_mg_dl (glucose mass/volume in serum or plasma)
- c_granulocytes_leukocytes_ratio (granulocytes/100 leukocytes in blood)
- c_hemoglobin_g_dl (hemoglobin mass/volume in blood)
- c_ldh_u_l (lactate dehydrogenase enzymatic activity/volume in serum or plasma)
- c_lymphocyte_10_9_l (lymphocytes #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_lymphocyte_leukocyte_ratio (lymphocytes/100 leukocytes in blood)
- c_monocytes_10_9_l (monocytes #/volume in blood)
- c_neutrophil_10_9_l (neutrophils #/volume in blood; used to compute neutrophil/lymphocyte ratio)
- c_platelets_10_9_l (platelets #/volume in blood)
- c_protein_g_l (protein mass/volume in Serum or Plasma)
- c_urea_nitrogen_mg_dl (urea nitrogen mass/volume in serum or plasma)

The function operates via the following logic:

1. Pull lab table and subset to patients included in inception cohort x.
2. Subset to lab of interest.

3. The lab date is derived as the date of the lab test itself.
4. Lab test results are categorized into "normal" and "abnormal" depending on if they fall within the provided reference.
5. The quantitative results are standardized to the same reference unit using conversion factors which are mapped in the `labs_mapping_edb4` table.
6. Measurements with missing lab test result (or implicitly missing unit due to prior conversion step) are discarded.
7. if `set_implausible_na` is set TRUE:
    - a lab test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the `labs_mapping_implausible_values` table.
8. All lab dates are restricted to the ones that fall within the measurement window specified in the `from` and `to` arguments for the lower and upper limits of the measurement window, respectively.
9. If multiple lab tests are available for a given patient, the prioritization step is carried out as follows:
    - For each patient, lab tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if `ties` = "higher") or ascending quantitative results to prioritize the lower measurement (if `ties` = "lower") measurements. The default is "lower".
    - The first row for each patient after the above sorting is selected
10. The newly created variables are named according to the standardized lab name and unit of measurement and joined back to `x`.

Value

x with all additional labs variables joined, that is:

- `c_(lab_name)_distance_(from)_(to)` - days from date of lab measurements to index date
- `c_(lab_name)_(unit)_(from)_(to)_cont` - quantitative lab result after unit harmonization/conversion

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb4_get_labs(
    cancer = "NSCLC",
    lab_name = "c_albumin_g_l",
    from = -90,
    to = 0
    )

## End(Not run)
```

edb4_get_os *Query overall survival outcome for a given inception cohort*

Description

Function queries mortality and other information to derive a righ-censored time to all-cause mortality endpoint

Usage

```
edb4_get_os(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = NULL,
  data_cut_off_date = lubridate::ymd("2024-09-30"),
  verbose = TRUE
)
```

Arguments

x	dataframe queried from edb4 with at least patient ids and index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where data/files are located
cancer	character, one of BC, CRC, MM or NSCLC
data_cut_off_date	date of database lock; the data cut-off for this delivery is Sep 30, 2024. This parameter can be changed if a grace period of x months before database lock is desired.
verbose	logical, print query progress and informative meta information

Details

The function queries and curates intention-to-treat (ITT) overall survival endpoint. The ITT follow-up time is defined as the time interval from index date (dt_index, including the index date) to the date of death (if death event occurred), or the last date of proof that the patient was alive at that time (date de-identified to week).

In EDB4, the granularity of the date of death variable is given on a month-level granularity. In these cases, the date of death is imputed to the mid/15th of the month. This can in rare circumstances lead to negative/implausible follow-up times if the index date is after the imputed date of death.

The function operates via the following logic:

1. Pull mortality details from the one-row-per-patient demographics table and subset to patients included in inception cohort x.
2. The date of death is imputed by default to the mid/15th of the month.
3. A patient's last structured activity date is provided as a structured field and defined as the last date of proof that the patient was alive at that time.

4. Mortality and last structured activity dates are joined to x and a binary death event variable is derived based on if a date of death is captured for a given patient.
5. The censoring date is determined as the date of death, if available. If no death date is available, the censoring date is determined as the last structured activity (see step 3) or at the data cut-off date (2024-09-30 is the default), whichever is earlier.
6. The follow-up (FU) time, defined as the time from index date (including index date) until censor date, is derived in days, months (= FU time in days/30.417) and years, depending on what the time scale should be for the respective analysis.
7. The newly created variables are joined back to x.

Value

x with all endpoint information joined, that is:

- death_itt, event indicator for all-cause mortality
- fu_itt_days, ITT follow-up time in days
- fu_itt_months, ITT follow-up time in months (i.e.,fu_itt_days / 30.417)
- fu_itt_years, ITT follow-up time in years

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb4_get_os(
    cancer = "NSCLC"
    )

## End(Not run)
```

---

edb4_get_vitals *Query vital sign measurements for a given cohort*

Description

Function queries vitals sign measurements

Usage

```
edb4_get_vitals(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = c("BC", "CRC", "MM", "NSCLC"),
  vital_name = NULL,
  from = -90,
  to = 0,
  ties = "lower",
  set_implausible_na = TRUE,
```

```
  label_name = FALSE,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB4 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index_date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of BC, CRC, MM or NSCLC
vital_name	character, curated name of vital sign measurement (see details)
from	integer, left boundary of vitals test measurement window relative to index date (e.g., -90, indicating vitals test should be measured not before 90 days before index date)
to	integer, right boundary of vitals test measurement window relative to index date (e.g., 0, indicating vitals test should be measured not later than the day of the index date (inclusive))
ties	character, in case of ties (two equi-distant measurements), should the "higher" or "lower" vitals test measurement be prioritized?
set_implausible_na	logical, should implausible values (outliers) be automatically be set NA (default is TRUE)? Lower and upper thresholds are documented in `vitals_mapping_edb4`
label_name	logical, should variable name carry information about the measurement window
verbose	logical, print query progress and informative meta information

Details

The function queries and cleans all available vital sign measurements. In detail, the function removes measurements that are character strings (e.g., "BMI < 21") and only considers quantitative results. The function further only selects measurements in a `from - to` measurement window relative to the index date. In case that there are multiple measurements in this window, the function uses the measurement that has the smallest absolute distance relative to the index date. If there are two equi-distant measurements, the `lower (default)` or `higher` measurement can be prioritized.

The available and supported vitals are:

- c_sbp (systolic blood pressure in mmHg)
- c_dbp (diastolic blood pressure in mmHg)
- c_bmi (body mass index in kg/m^2 directly measured or derived from weight/height^2)
- c_bsa (body surface area in m^2)
- c_height (height in m)
- c_hr (heart rate/pulse in beats/min)
- c_oxygen (oxygen saturation; taken from O2 sat and pulse oximetry)
- c_pain (pain scale in ?)
- c_resp (respiration in breaths/min)
- c_temp (temperature in fahrenheit)
- c_weight (weight in kg)

The function operates via the following logic:

1. Pull vitals table and subset to patients included in inception cohort `x`.
2. Subset to vitals measurement of interest.
3. The quantitative results are standardized to the same reference unit using conversion factors which are mapped in the `vitals_mapping_edb4` table.
4. The vitals test date is derived as the date of the vitals test date itself.
5. Measurements with missing vital test result are discarded.
6. Some vital measurements are encoded as characters/strings and are cleaned/converted to numeric values.
7. if `set_implausible_na` is set TRUE:
   - a vitals test measurement is discarded if the quantitative test result falls outside the physiologically plausible ranges defined in the `vitals_mapping_implausible_values` table.
8. All vital dates are restricted to the ones that fall within the measurement window specified in the `from` and `to` arguments for the lower and upper limits of the measurement window, respectively.
9. If multiple vitals tests are available for a given patient, the prioritization step is carried out as follows:
   - For each patient, vitals tests are arranged by ascending absolute distance to the index date and descending quantitative results to prioritize the higher measurement (if `ties` = "higher") or ascending quantitative results to prioritize the lower measurement (if `ties` = "lower") measurements. The default is "lower".
   - The first row for each patient after the above sorting is selected
10. The newly created variables are named according to the standardized vital test name and unit of measurement and joined back to `x`.

Value

x with all additional vitals test variables joined, that is:

- `c_(vital_name)_distance_(from)_(to)` - days from date of vital sign measurements to index date
- `c_(vital_name)_(unit)_(from)_(to)_cont` - quantitative vital sign measurement

Note: if `from` or `to` is a negative integer, the resulting covariate name will contain a "min" for "minus" preceding the integer to comply with the tidy variable naming rules

Examples

```
## Not run:
library(encore.io)

ard <- x |>
  edb4_get_vitals(
    cancer = "NSCLC",
    vital_name = "c_bmi",
    from = -90,
    to = 0
    )

## End(Not run)
```

edb4_query_ropro *Query and curate all relevant ROPRO variables*

Description

This function queries, cleans and transforms all necessary covariates needed to compute the ROPRO prognostic score in EDB4.

Usage

```
edb4_query_ropro(
  x = NULL,
  index_date = "dt_index",
  path = Sys.getenv("path_edb4"),
  cancer = NULL,
  from = -90,
  to = 0,
  verbose = TRUE
)
```

Arguments

x	dataframe queried from EDB4 with at least patient ids and therapy index date of inception cohort
index_date	character, variable/column name with the patient's index date, default is dt_index
path	character string, path to directory where EDB4 data/files are located
cancer	character, one of "BC", "CRC", "MM" or "NSCLC"
from	integer, left boundary of measurement window for time-dependent variables relative to index date (e.g., -90, indicating variables should be measured not before 90 days before index date)
to	integer, right boundary of measurement window for time-dependent variables relative to index date (e.g., 0, indicating variables should be measured not later than the day of the index date (inclusive))
verbose	logical, print progress of query

Details

Wrapper around major functions to query required covariates to compute ROPRO. Selected covariates are log transformed or log-log transformed. More details, see Becker, Weberpals, et al., Ann Oncol 2020.

Value

x with all required ROPRO covariates joined. All general and cancer type-specific (as specified in `cancer` argument) covariates are returned.

#' Note that:

- covariates for EarlyBreast are identical to the the general pan-tumor ROPRO, just the weights are different. That means, if `cancer = "BC"` is specified, the function also returns covariates for the metastatic breast cancer-specific ROPRO.

Examples

```
## Not run:
library(encore.io)

x_ropro <- x |>
  edb4_query_ropro(
    index_date = "dt_index",
    path = Sys.getenv("path_edb4"),
    cancer = "NSCLC",
    from = -90,
    to -0,
    verbose = TRUE
    )


## End(Not run)
```

`ess` *Calculates the averaged effective sample sizes for mimids/wimids objects*

Description

[Deprecated] This functions was deprecated in favor of `ESS`.

Calculates the averaged effective sample sizes for imputed and matched or weighted datasets resulting from `MatchThem::matchthem(mimids objects)` or `MatchThem::weightthem(wimids objects)`

Usage

```
ess(object = NULL, decimals = 2)
```

Arguments

`object`	mimids or data.frame object from complete(..., action = 'long', all = TRUE, ...)
`decimals`	decimals to round for averaged effective sample size (default is 2 decimals)

Details

Matching or weighting across imputed datasets with partially observed covariates leads to slightly differenmatched or weighted cohorts with different sample sizes for each imputed dataset.

To account for this in the descriptive reporting of the effective sample size used in our analysis, this function computes the averaged effective sample sizes across all matched or weighted cohorts.

This function is a wrapper around the `bal.tab` function of the cobalt R package which performs these computations. For more information on how the effective sample sizes are computed.

Value

tibble with sample size information by treatment indicator

See Also

bal.tab

https://ngreifer.github.io/cobalt/reference/bal.tab.html

Examples

```
## Not run:

if(require("smdi") & require("MatchThem")){

  library(smdi)
  library(mice)
  library(MatchThem)
  library(encore.io)

  mids <- mice(smdi_data, printFlag = F)

  fit <- as.formula(exposure ~ age_num + female_cat + ecog_cat + egfr_cat + pdl1_num)
  wimids <- weightthem(fit, mids)

  ess(wimids, decimals = 1)

}


## End(Not run)
```

---

gt_tbl_compact *Utility function for a more compact look of gt tables*

Description

[Experimental]

Is a convenience utility function to make gt tables look more compact as compared to the default size and spacing.

#' @seealso gt theme_gtsummary_compact

Usage

```
gt_tbl_compact(gt_tbl = NULL, font_size = 13)
```

Arguments

gt_tbl	gt table object
font_size	numeric, desired font size (default is 13)

Value

gt table in compact format

Examples

```
library(encore.io)
library(gt)

head(iris) |>
 gt() |>
 gt_tbl_compact()
```

`icd_metastases` *ICD-10-CM to ICD-9-CM equivalence crosswalk table*

Description

ICD-10-CM to ICD-9-CM equivalence crosswalk table

Usage

```
icd_metastases
```

Format

`icd_metastases`:

ICD-10-CM to ICD-9-CM equivalence crosswalk for secondary malignancy codes (C77x, C78x, C79x) taken from NBER

icd10cm ICD-10-CM code

icd9cm Mapped ICD-9-CM code

approximate Approximate map flag - identifies entries where the complete meaning of the source system code and that of the target system code are not considered equivalent

no_map No map flag - distinguishes entries where the source system code has at least one translation from entries where the source system code has no target system translation

sites Organ site description ...

Source

https://www.nber.org/research/data/icd-9-cm-and-icd-10-cm-and-icd-10-pcs-crosswalk-or-general-e

---

`imputation_workflow` *Streamlines the imputation workflow*

Description

This function streamlines the imputation workflow from an eligible cohort with missings to an imputed mids object with only a subset of the key covariates and computed ropro

Usage

```
imputation_workflow(
  ard_eligible = NULL,
  database = NULL,
  cancer = NULL,
  covars_for_imputation = NULL
)
```

Arguments

ard_eligible	data frame with all eligible patients and covariates needed for imputation
database	character, which database is used ("edb1", "edb2", "edb3" or "edb4")
cancer	character, which cancer is investigated ("aNSCLC", "MetastaticBreast", "EarlyBreast", "MetastaticCRC", "MultipleMyeloma")
covars_for_imputation	character vector with covariates for imputation (should include covariates for propensity score, treatment, outcome and optionally other auxiliary covariates)

Details

...

Value

imputed mids object with ROPRO

Examples

```
## Not run:
if(require("smdi")){

  library(encore.io)

  mids_data <- imputation_workflow(
    ard_eligible,
    database = "edb1",
    cancer = "aNSCLC",
    covars_for_imputation = c("dem_age_index", "dem_sex")
    )

 }

## End(Not run)
```

---

km_pooling *Pooled Kaplan-Meier estimate and survival curve*

Description

Computes pooled median survival Kaplan-Meier estimates using Rubin's rule and outputs corresponding Kaplan-Meier curve across imputed and matched/weighted datasets

Usage

```
km_pooling(
  object = NULL,
  surv_formula = as.formula(survival::Surv(eventtime, status) ~ exposure)
)
```

Arguments

object	imputed and matched (mimids) or weighted (wimids) object
surv_formula	specification of survival model formula to be fitted

Details

The function requires an object of class mimids or wimids, which is the output of a workflow that requires imputing multiple (m) datasets using mice or amelia and matching or weighting each imputed dataset via the MatchThem package (see examples).

The function fits the pre-specied survfit model (`surv_formula`) to compute survival probabilities at each individual time point according to the Kaplan-Meier method. For matched and weighted datasets, weights, cluster membership (matching only) and robust variance estimates are considered in the `survfit` call by default.

Since survival probabilities typically don't follow normal distributions, these need to be transformed to approximate normality first before pooling across imputed datasets and time points. To that end, survival probabilities are first transformed using a complementary log-log transformation (`log(-log(1-pr(surv)))`) as recommended by multiple sources (Marshall, Billingham, and Bryan (2009)).

To pool the transformed estimates across imputed datasets and time points, the pool.scalar function is used to apply Rubin's rule to combine pooled estimates (qbar) according to formula (3.1.2) Rubin (1987) and to compute the corresponding total variance (t) of the pooled estimate according to formula (3.1.5) Rubin (1987).

The pooled survival probabilities are then back-transformed via `1-exp(-exp(qbar))` for pooled survival probability estimates and `1-exp(-exp(qbar +/- 1.96*sqrt(t)))` for lower and upper 95% confidence intervals. As the formula indicates, the pooled standard error is computed as the square root of the total variance. The vertically stacked table with transformed and backtransformed estimates is returned with the `km_survival_table` table.

Finally, the median survival time is extracted from the `km_survival_table` table by determining the time the survival probability drops below .5 for the first time. For this a sub-function of Terry M. Therneau's `print.survfit` function is used. Therneau also considers some edge cases/nuisances (x = time, y = surv):

- Nuisance 1: if one of the y's is exactly .5, we want the mean of the corresponding x and the first x for which y<.5. We need to use the equivalent of all.equal to check for a .5 however: survfit(Surv(1:100)~1) gives a value of .5 + 1.1e-16 due to roundoff error.
- Nuisance 2: there may by an NA in the y's
- Nuisance 3: if no y's are <=.5, then we should return NA
- Nuisance 4: the obs (or many) after the .5 may be censored, giving a stretch of values = .5 +- epsilon

More references:

- https://stefvanbuuren.name/fimd/sec-pooling.html
- https://link.springer.com/article/10.1007/s10198-008-0129-y
- https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-015-0048-4

Value

list with pooled median survival estimate and pooled Kaplan-Meier curve `km_median_survival`:

- strata = stratum
- t_median = median survival time
- t_lower = lower 95% CI of median survival time
- t_upper = upper 95% CI of median survival time

`km_survival_table`:

- strata = stratum
- time = observed time point
- m = number of imputed datasets
- qbar = pooled univariate estimate of the complementary log-log transformed survival probabilities, see formula (3.1.2) Rubin (1987)
- t = total variance of the pooled univariate estimate of the complementary log-log transformed survival probabilities, formula (3.1.5) Rubin (1987)
- se = total standard error of the pooled estimate (derived as sqrt(t))
- surv = back-transformed pooled survival probability
- lower = Wald-type lower 95% confidence interval of back-transformed pooled survival probability
- upper = Wald-type upper 95% confidence interval of back-transformed pooled survival probability

`km_plot`: ggplot2 object with Kaplan-Meier curve

See Also

`survfit` `pool.scalar` `matchthem` `weightthem`

Examples

```
if(require("MatchThem")){

  library(smdi)
  library(mice)
  library(MatchThem)
  library(encore.io)

  # impute data
  set.seed(42)
  mids <- mice(smdi_data[1:500, ], m = 2, printFlag = FALSE)

  # fit a propensity score model
  fit <- as.formula(exposure ~ age_num + female_cat + ecog_cat + egfr_cat + pdl1_num)

  # weight (or alternatively match) patients within each imputed dataset
  wimids <- weightthem(
    formula = fit,
    datasets = mids,
    approach = "within",
    method = "glm",
```

```
  estimand = "ATO"
  )

# specificy survival model
km_fit <- as.formula(survival::Surv(eventtime, status) ~ exposure)

# estimate and pool median survival times and Kaplan-Meier curve
km_out <- km_pooling(
  object = wimids,
  surv_formula = km_fit
  )

# median survival time
km_out$km_median_survival

# KM curve
km_out$km_plot

}
```

`labs_mapping_edb1` *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
labs_mapping_edb1
```

Format

`labs_mapping_edb1`**:**

Lookup table for unique lab test and unit combinations in EDB1

test Test name as found in data

testunitscleaned Harmonized test unit

n N observed

ropro Is vital sign deemed to be prognostic

lab_name_clean Harmonized variable name

unit_clean Harmonized variable unit

lower_implausible_li Lower implausible threshold

upper_implausible_li Upper implausible threshold ...

labs_mapping_edb2 *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
labs_mapping_edb2
```

Format

`labs_mapping_edb2`:
Lookup table for unique lab test and unit combinations in EDB2

database Database
lab_name Test name as found in data
unit Unit for test name as found in data
n N observed
ropro Is vital sign deemed to be prognostic
lab_name_clean Harmonized variable name
unit_clean Harmonized variable unit
conversion_factor Conversion factor to multiply with to get standardized unit
comment Comment
lower_implausible_li Lower implausible threshold
upper_implausible_li Upper implausible threshold ...

labs_mapping_edb3 *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
labs_mapping_edb3
```

Format

`labs_mapping_edb3`:
Lookup table for unique lab test and unit combinations in EDB3

database Database
lab_name Test name as found in data
unit Unit for test name as found in data
n N observed

ropro Is vital sign deemed to be prognostic
lab_name_clean Harmonized variable name
unit_clean Harmonized variable unit
comment Comment
lower_implausible_li Lower implausible threshold
upper_implausible_li Upper implausible threshold ...

`labs_mapping_edb4` *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
labs_mapping_edb4
```

Format

`labs_mapping_edb4`**:**
Lookup table for unique lab test and unit combinations in EDB4

database Database
lab_name Test name as found in data
n N observed
unit Unit for test name as found in data
ropro Is vital sign deemed to be prognostic
lab_name_clean Harmonized variable name
unit_clean Harmonized variable unit
conversion_factor Conversion factor to multiply with to get standardized unit
comment Comment
lower_implausible_li Lower implausible threshold
upper_implausible_li Upper implausible threshold ...

`labs_mapping_implausible_values`
Threshold table for plausible value ranges for selected labs

Description

Threshold table for plausible value ranges for selected labs

Usage

```
labs_mapping_implausible_values
```

Format

`labs_mapping_implausible_values`**:**

Thresholds for plausible value ranges for selected labs in a given unit; list was developed as part of the ENCORE project together with physicians from Dana-Farber Cancer Institute and MassGeneralBrigham

lab_name_clean Harmonized lab variable name as derived in the `fh_get_labs()` function

lower_implausible_limit Lower implausible threshold

upper_implausible_limit Upper implausible threshold ...

`n_fmt` *Quickly format numbers*

Description

format raw numeric numbers into formatted characters including large decimal "," and small decimal "."

Usage

```
n_fmt(x, n_digits = 2)
```

Arguments

`x`	number
`n_digits`	integer, number of digits after comma

Details

...

Value

a character of the formatted numbers

Examples

```
## Not run:
library(encore.io)
n_fmt(12345678)

## End(Not run)
```

power_survival *Power analysis for proportional hazards models*

Description

[Deprecated] This functions was deprecated in favor of `gsDesign::nEvents`.

Function computes implementations of sample-Size Formula for the Proportional-Hazards Regression Model for the statistical power of a two arm treatment comparison as described by David A. Schoenfeld (Biometrika 1983)

Usage

```
power_survival(
  beta = NULL,
  alpha = NULL,
  p_exposed = NULL,
  n_events = NULL,
  hr = NULL
)
```

Arguments

`beta`	numeric, beta percentiles of the normal distribution (type II error rate;)
`alpha`	numeric, 1-alpha percentiles of the normal distribution (type I error rate)
`p_exposed`	numeric, proportion of exposed patients
`n_events`	numeric, number of events
`hr`	numeric, hazard ratio

Details

One parameter can be left undefined (except `p_exposed`) which is the computed using an implementation of David A. Schoenfeld's (Biometrika 1983) formula.

Value

integer representing the undefined parameter

See Also

Schoenfeld DA. Sample-size formula for the proportional-hazards regression model. Biometrics 1983;39:499-503.

https://www.jstor.org/stable/2531021

Examples

```
library(encore.io)

power_survival(
  alpha = 0.05,
  beta = 0.2,
```

```
p_exposed = 0.5,
hr = 0.8
)
```

`ps_balance_plot` *Computes and visualizes the overlap for distance measures between exposure groups*

Description

[Deprecated] This functions was deprecated in favor of `bal.plot`. Calculates the the overlap for distance measures (e.g., propensity or prognostic scores) between exposure groups for multiple imputed and matched datasets.

Usage

```
ps_balance_plot(
  object = NULL,
  exposure = "treat",
  weights = "weights",
  ps = "distance"
)
```

Arguments

`object`	mimids or data.frame object from complete(..., action = 'long', all = TRUE, ...)
`exposure`	character, quoted name of the exposure/treatment variable
`weights`	character, quoted name of the variable indicating the matching weights (usually 0: unmatched and 1: matched)
`ps`	character, quoted name of the variable with the distance measure (e.g., propensity score)

Details

The object input needs to be a mimids object or a data.frame object coming from MatchThem::matchthem(). If the mimids object is already converted to a long data.frame of stacked imputed datasets, the MatchThem::complete() function needs to be completed using action = "long" and all = TRUE arguments.

The function then creates two stacked datasets (unmatched/all and matched only) patients and combines the propensity score across all imputed datasets into a single graph.

Value

ggplot object

Examples

```
## Not run:
library(encore.io)

ps_balance_plot(
  object = edb1_mimids,
  exposure = "treat",
  ps = "distance"
  )


## End(Not run)
```

`qc_assertive_line_check` *Assertive line of therapy checks*

Description

Assertive line of therapy checks to make sure that patients in an advanced line of treatment also received a previous line. Important: input dataframe needs to have a one line per patient per line of therapy format.

Usage

```
qc_assertive_line_check(
  data = NULL,
  id_col = NULL,
  linenumber_col = NULL,
  linename_col = NULL
)
```

Arguments

`data`	dataframe including line number and line treatment
`id_col`	quoted character specifying the column name of the patient identifier
`linenumber_col`	quoted character specifying the column name of the line number
`linename_col`	quoted character specifying the column name of the line name/treatment

Details

Data quality check, not to use for analysis.

Value

a table/dataframe with the logical results (TRUE/FALSE) of the assertive checks.

Examples

```
## Not run:
library(encore.io)


## End(Not run)
```

`re_weight` *Custom function to perform matching/weighting and re-weighting to match a target patient population*

Description

[Experimental] Performs re-weighting based on the `anesrake` function.

Usage

```
re_weight(x, targets = NULL, matching_weighting = NULL, ...)
```

Arguments

`x`	data.frame or mild object/list of data.frames if used in combination with lapply
`targets`	named list of all target values for the raking procedure (see `anesrake`)
`matching_weighting`	character, one of "matching" or "weighting"
`...`	other arguments and specifications to propagate on to `matchit` or `weightit`, depending on chosen method (`matching_weighting`).

Details

This function is a wrapper for `matchit` and `weightit`in combination with `anesrake`.

The function performs any matching algorithm supplied in `matchit` or any weighting algorithm in `weightit`. The specific arguments can be propagated to the respective functions using the `...` argument.

If nothing is specified for `targets`, the function will simply return the `matchit` or `weightit` object.

If a list of target distributions is specified for `targets`, the function will perform a corresponding re-weighting of matched patients or patients with weights greater than 0, respectively.

In case of an already weighted datasets (e.g., via propensity score-derived weights), those weights are accounted for (`weightvec` argument) in the `anesrake` call.

Value

an object of type `matchit` or `weightit` with included sampling weights (`s.weights`) if re-weighting was performed.

See Also

`matchit` `weightit` `anesrake`

Examples

```
if(require("MatchThem") & require("mice")){

library(encore.io)
library(dplyr)
library(mice)
library(MatchThem)

data_miss <- simulate_flaura(
  n_total = 3500,
  seed = 41,
  include_id = FALSE,
  imposeNA = TRUE,
  propNA = .33
  ) |>
 # anesrake works best with factor variables
 # convert c_smoking_history into a factor
 mutate(c_smoking_history = factor(ifelse(c_smoking_history == TRUE, "Current/former", "Never")))

 # impute data
 data_imp <- mice(
   parallelseed = 42,
   n.core = 7,
   data = data_miss,
   m = 5,
   print = FALSE
   )

 # define covariates for propensity score model
 covariates <- data_miss |>
   select(starts_with("c_"), starts_with("dem_")) |>
   colnames()

 # define propensity score model
 ps_form <- as.formula(paste("treat ~", paste(covariates, collapse = " + ")))

 # create a mild object containing lists of data.frames
 data_mild <- mice::complete(data = data_imp, action = "all", include = FALSE)

 smoker_target <- c(.35, .65)
 names(smoker_target) <- c("Current/former", "Never")

 # summarize target distributions in a named list vector --------------
 targets <- list(smoker_target)
 names(targets) <- c("c_smoking_history")

 # create a mild object containing lists of data.frames
 data_mild <- mice::complete(data = data_imp, action = "all", include = FALSE)

 # call re-weight
 matchit_out_list <- lapply(
  X = data_mild,
  FUN = re_weight,
  targets = targets,
  matching_weighting = "matching",
  # arguments passed on to matchit
```

```
    formula = ps_form,
    ratio = 1,
    method = "nearest",
    distance = "glm",
    link = "logit",
    caliper = 0.01,
    replace = FALSE
    )

  # convert the output back into a mimids object
  data_mimids_from_function <- MatchThem::as.mimids(
    x = matchit_out_list,
    datasets = data_imp
    )

  # print
  data_mimids_from_function
}
```

`ropro_aNSCLC_covars` *aNSCLC-specific ROPRO covariate vector*

Description

aNSCLC-specific ROPRO covariate vector

Usage

```
ropro_aNSCLC_covars
```

Format

`ropro_aNSCLC_covars`**:**
Covariate vector with harmonized covariate names for aNSCLC-specific ROPRO

`ropro_covars_log_log_transform`
Covariate vector of covariates that are log-log-transformed

Description

Covariate vector of covariates that are log-log-transformed

Usage

```
ropro_covars_log_log_transform
```

Format

`ropro_covars_log_log_transform`**:**
Covariate vector of covariates that are log-log-transformed

ropro_covars_log_transform

Covariate vector of covariates that are log-transformed

Description

Covariate vector of covariates that are log-transformed

Usage

```
ropro_covars_log_transform
```

Format

`ropro_covars_log_transform`:
Covariate vector of covariates that are log-transformed

ropro_earlyBreast_covars

early Breast-specific ROPRO covariate vector

Description

early Breast-specific ROPRO covariate vector

Usage

```
ropro_earlyBreast_covars
```

Format

`ropro_earlyBreast_covars`:
Covariate vector with harmonized covariate names for early Breast-specific ROPRO

ropro_mBreast_covars

metastatic Breast-specific ROPRO covariate vector

Description

metastatic Breast-specific ROPRO covariate vector

Usage

```
ropro_mBreast_covars
```

Format

`ropro_mBreast_covars`:
Covariate vector with harmonized covariate names for metastatic Breast-specific ROPRO

ropro_mCRC_covars *metastatic CRC-specific ROPRO covariate vector*

Description

metastatic CRC-specific ROPRO covariate vector

Usage

```
ropro_mCRC_covars
```

Format

`ropro_mCRC_covars`:
Covariate vector with harmonized covariate names for metastatic CRC-specific ROPRO

ropro_MM_covars *MM-specific ROPRO covariate vector*

Description

MM-specific ROPRO covariate vector

Usage

```
ropro_MM_covars
```

Format

`ropro_MM_covars`:
Covariate vector with harmonized covariate names for MMspecific ROPRO

ropro_pan_tumor_covars *Pan-tumor ROPRO covariate vector*

Description

Pan-tumor ROPRO covariate vector

Usage

```
ropro_pan_tumor_covars
```

Format

`ropro_pan_tumor_covars`:
Covariate vector with harmonized covariate names for pan-tumor ROPRO

ropro_pan_tumor_covars_categorical

Pan-tumor ROPRO covariate vector (modified for categorical covariates)

Description

Pan-tumor ROPRO covariate vector (modified for categorical covariates)

Usage

```
ropro_pan_tumor_covars_categorical
```

Format

`ropro_pan_tumor_covars_categorical`:
Covariate vector with harmonized covariate names for pan-tumor ROPRO. Age, sex and ECOG are included as a trurly categorical (factor) variable

simulate_flaura

Simulates and artifical FLAURA EHR-derived dataset

Description

Parameterized function to quickly create an artificial FLAURA EHR-derived analytic cohort for analytic code development.

Usage

```
simulate_flaura(
  n_total = 3500,
  seed = 42,
  include_id = TRUE,
  imposeNA = TRUE,
  propNA = NULL
)
```

Arguments

n_total	integer, number of total patients
seed	integer, seed for reproducibility
include_id	logical, include a generated patientid variable
imposeNA	logical, set covariates to missing
propNA	numeric, proportion of missingness, needs to be between 0 and 1

Details

...

Value

data frame with simulated analytic cohort

Examples

```
## Not run:
library(encore.io)

data_miss <- simulate_flaura(
  n_total = 3500,
  seed = 41,
  include_id = FALSE,
  imposeNA = TRUE,
  propNA = .33
  )

head(data_miss)


## End(Not run)
```

---

state_region_mapping *A mapping between US States and geographical regions*

Description

A mapping between US States and geographical regions

Usage

```
state_region_mapping
```

Format

`state_region_mapping`**:**
A mapping between US States and geographical regions

state US State
dem_region Mapped geographical region ...

---

vitals_mapping_edb1 *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
vitals_mapping_edb1
```

Format

`vitals_mapping_edb1`:
Lookup table for unique lab test and unit combinations in EDB1

test Test name as found in data
testunitscleaned Harmonized test unit
n N observed
ropro Is vital sign deemed to be prognostic
vital_name_clean Harmonized variable name
unit_clean Harmonized variable unit
conversion_factor Factor to convert vital sign measurement to harmonized unit
comment Comment
lower_implausible_li Lower implausible threshold
upper_implausible_li Upper implausible threshold ...

---

vitals_mapping_edb2 *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
vitals_mapping_edb2
```

Format

`vitals_mapping_edb2`:
Lookup table for unique lab test and unit combinations in EDB2

vital_name Vital test name as found in data
vital_name_clean Harmonized variable name
lower_implausible_li Lower implausible threshold
upper_implausible_li Upper implausible threshold ...

---

vitals_mapping_edb3 *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
vitals_mapping_edb3
```

Format

`vitals_mapping_edb3`:

Lookup table for unique lab test and unit combinations in EDB3

database Database

test_category Test category

vital_name Test name as found in data

vital_unit Harmonized test unit

n N observed

ropro Is vital sign deemed to be prognostic

vital_name_clean Harmonized variable name

unit_clean Harmonized variable unit

conversion_factor Factor to convert vital sign measurement to harmonized unit

comment Comment

lower_implausible_li Lower implausible threshold

upper_implausible_li Upper implausible threshold ...

`vitals_mapping_edb4` *Lookup table for unique lab test observations*

Description

Lookup table for unique lab test observations

Usage

```
vitals_mapping_edb4
```

Format

`vitals_mapping_edb4`:

Lookup table for unique lab test and unit combinations in EDB4

vital_name Test name as found in data

vital_name_clean Harmonized variable name

unit_clean Harmonized variable unit

lower_implausible_li Lower implausible threshold

upper_implausible_li Upper implausible threshold ...

`vitals_mapping_implausible_values` *Threshold table for plausible value ranges for selected vital slign measurements*

Description

Threshold table for plausible value ranges for selected vital slign measurements

Usage

```
vitals_mapping_implausible_values
```

Format

`vitals_mapping_implausible_values`**:**

Thresholds for plausible value ranges for selected labs in a given unit; list was developed as part of the ENCORE project together with physicians from Dana-Farber Cancer Institute and MassGeneralBrigham

vital_name_clean Harmonized vitals variable name as derived in the `fh_get_vitals()` function

lower_implausible_limit Lower implausible threshold

upper_implausible_limit Upper implausible threshold ...

Index

∗ **datasets**
edb1_cohorts, 7
icd_metastases, 86
labs_mapping_edb1, 90
labs_mapping_edb2, 91
labs_mapping_edb3, 91
labs_mapping_edb4, 92
labs_mapping_implausible_values, 92
ropro_aNSCLC_covars, 99
ropro_covars_log_log_transform, 99
ropro_covars_log_transform, 100
ropro_earlyBreast_covars, 100
ropro_mBreast_covars, 100
ropro_mCRC_covars, 101
ropro_MM_covars, 101
ropro_pan_tumor_covars, 101
ropro_pan_tumor_covars_categorical, 102
state_region_mapping, 103
vitals_mapping_edb1, 103
vitals_mapping_edb2, 104
vitals_mapping_edb3, 104
vitals_mapping_edb4, 105
vitals_mapping_implausible_values, 106

anesrake, *97*

bal.plot, *95*
bal.tab, *84, 85*

c_statistics, 5
create_table1, 4

edb1_3_4_compute_ropro, 6
edb1_cohorts, 7
edb1_get_biomarker, 7
edb1_get_demographics, 10
edb1_get_diagnosis_heme, 12
edb1_get_diagnosis_solid, 14
edb1_get_ecog, 16
edb1_get_histology, 18
edb1_get_labs, 19
edb1_get_os, 22
edb1_get_vitals, 24
edb1_query_ropro, 26
edb2_assign_date, 28
edb2_compute_ropro, 29
edb2_get_biomarker, 30
edb2_get_demographics, 32
edb2_get_diagnosis_heme, 34
edb2_get_diagnosis_solid, 35
edb2_get_ecog, 38
edb2_get_histology, 40
edb2_get_labs, 41
edb2_get_os, 44
edb2_get_vitals, 46
edb2_path_helper, 48
edb2_query_ropro, 49
edb3_get_biomarker, 50
edb3_get_demographics, 52
edb3_get_diagnosis_solid, 53
edb3_get_ecog, 55
edb3_get_histology, 56
edb3_get_labs, 57
edb3_get_os, 60
edb3_get_vitals, 61
edb3_query_ropro, 63
edb4_get_biomarker, 65
edb4_get_demographics, 67
edb4_get_diagnosis_heme, 69
edb4_get_diagnosis_solid, 70
edb4_get_ecog, 72
edb4_get_histology, 74
edb4_get_labs, 76
edb4_get_os, 79
edb4_get_vitals, 80
edb4_query_ropro, 83
ESS, *84*
ess, 84

gt, *85*
gt_tbl_compact, 85

icd_metastases, 86
imputation_workflow, 86

km_pooling, 87

labs_mapping_edb1, 90
labs_mapping_edb2, 91
labs_mapping_edb3, 91
labs_mapping_edb4, 92
labs_mapping_implausible_values, 92

matchit, *97*
matchthem, *89*

n_fmt, 93

pool.scalar, *89*
power_survival, 94
print.survfit, *88*
ps_balance_plot, 95

qc_assertive_line_check, 96

re_weight, 97
ropro_aNSCLC_covars, 99
ropro_covars_log_log_transform, 99
ropro_covars_log_transform, 100
ropro_earlyBreast_covars, 100
ropro_mBreast_covars, 100
ropro_mCRC_covars, 101
ropro_MM_covars, 101
ropro_pan_tumor_covars, 101
ropro_pan_tumor_covars_categorical,
        102

simulate_flaura, 102
state_region_mapping, 103
survfit, *88, 89*

tbl_summary, *4*
theme_gtsummary_compact, *85*

vitals_mapping_edb1, 103
vitals_mapping_edb2, 104
vitals_mapping_edb3, 104
vitals_mapping_edb4, 105
vitals_mapping_implausible_values, 106

weightit, *97*
weightthem, *89*

Supplementary Material last updated: 2025-03-30