Package ‘CodelistGenerator’

July 12, 2024

Title  Identify Relevant Clinical Codes and Evaluate Their Use

Version  3.1.0

Description  Generate a candidate code list for the Observational Medical Outcomes Partnership (OMOP) common data model based on string matching. For a given search strategy, a candidate code list will be returned.

License  Apache License (>= 2)

Encoding  UTF-8

RoxygenNote  7.3.2

Depends  R (>= 3.5.0)

Imports  checkmate (>= 2.0.0), DBI (>= 1.1.0), dplyr (>= 1.1.0), magrittr (>= 2.0.0), omopgenerics (>= 0.2.2), rlang (>= 1.0.0), glue (>= 1.5.0), stringr (>= 1.4.0), tidyselect (>= 1.2.0), tidyr (>= 1.2.0), cli (>= 3.1.0), purrr, lubridate, PatientProfiles (>= 1.1.0), RJSONIO, vctrs, visOmopResults (>= 0.3.0), lifecycle

Suggests  covr, duckdb, CDMConnector (>= 1.3.0), knitr, rmarkdown, testthat (>= 3.0.0), RPostgres, odbc, spelling, tibble, gt, flextable

Config/testthat/edition  3

Config/testthat/parallel  true

VignetteBuilder  knitr

URL  https://darwin-eu.github.io/CodelistGenerator/

Language  en-US

LazyData  true

NeedsCompilation  no

Author  Edward Burn [aut, cre] (<https://orcid.org/0000-0002-9286-1128>), Marti Catala [ctb] (<https://orcid.org/0000-0003-3308-9905>), Xihang Chen [ctb] (<https://orcid.org/0000-0001-8112-8958>), Mike Du [ctb] (<https://orcid.org/0000-0002-9517-8834>), Danielle Newby [ctb] (<https://orcid.org/0000-0002-3001-1478>)
Maintainer Edward Burn <edward.burn@ndoms.ox.ac.uk>
Repository CRAN
Date/Publication 2024-07-12 07:10:02 UTC

Contents

codesFromCohort .............................................. 3
codesFromConceptSet ....................................... 3
codesInUse .................................................... 4
compareCodelists ............................................. 5
doseFormToRoute ............................................. 6
getATCCodes ................................................... 6
getCandidateCodes .......................................... 7
getConceptClassId ........................................... 8
getDescendants ................................................. 9
getDomains .................................................... 10
getDoseForm .................................................. 11
getDoseUnit ................................................... 11
getDrugIngredientCodes ..................................... 12
getICD10StandardCodes ..................................... 13
getMappings ................................................... 14
getRelationshipId ............................................ 15
getRouteCategories ......................................... 16
getVocabularies .............................................. 16
getVocabVersion .............................................. 17
mockVocabRef .................................................. 17
restrictToCodesInUse ........................................ 18
sourceCodesInUse ............................................. 19
stratifyByConcept ........................................... 20
stratifyByDoseUnit .......................................... 20
stratifyByRouteCategory .................................... 21
subsetOnDoseUnit ............................................. 21
subsetOnRouteCategory ...................................... 22
subsetToCodesInUse ......................................... 22
summariseAchillesCodeUse .................................. 23
summariseCodeUse ............................................ 24
summariseCohortCodeUse .................................... 25
summariseOrphanCodes ....................................... 27
tableAchillesCodeUse ....................................... 28
tableCodeUse .................................................. 29
tableCohortCodeUse ......................................... 30
tableOrphanCodes ............................................. 31

Index 34
codesFromCohort  Get concept ids from a provided path to cohort json files

Description
Get concept ids from a provided path to cohort json files

Usage
codesFromCohort(path, cdm, type = c("codelist"))

Arguments
path  Path to a file or folder containing JSONs of cohort definitions
cdm  A cdm reference created with CDMConnector
type  Can be "codelist", "codelist_with_details", or "concept_set_expression"

Value
Named list with concept_ids for each concept set

codesFromConceptSet  Get concept ids from a provided path to json files

Description
Get concept ids from a provided path to json files

Usage
codesFromConceptSet(path, cdm, type = c("codelist"))

Arguments
path  Path to a file or folder containing JSONs of concept sets
cdm  A cdm reference created with CDMConnector
type  Can be "codelist", "codelist_with_details", or "concept_set_expression"

Value
Named list with concept_ids for each concept set
## Not run:
cdm <- mockVocabRef("database")
x <- codesFromConceptSet(cdm = cdm,
  path = system.file(package = "CodelistGenerator",
  "concepts_for_mock"))
x
CDMConnector::cdmDisconnect(cdm)
## End(Not run)

codesInUse

Use achilles counts to get codes used in the database

### Usage

codesInUse(
  cdm,
  minimumCount = 0,
  table = c("condition_occurrence", "device_exposure", "drug_exposure", "measurement",
  "observation", "procedure_occurrence", "visit_occurrence")
)

### Arguments

cdm cdm_reference via CDMConnector
minimumCount Any codes with a frequency under this will be removed.
table cdm table

### Value

A list of integers indicating codes being used in the database.

### Examples

## Not run:
cdm <- mockVocabRef("database")
x <- codesInUse(cdm = cdm)
x
CDMConnector::cdmDisconnect(cdm)
## End(Not run)
compareCodelists  Compare two codelists

Description

Compare two codelists

Usage

compareCodelists(codelist1, codelist2)

Arguments

codelist1  Output of getCandidateCodes
codelist2  Output of getCandidateCodes

Value

tibble

Examples

```r
## Not run:
cdm <- mockVocabRef()
codes1 <- getCandidateCodes(
  cdm = cdm,
  keywords = "Arthritis",
  domains = "Condition",
  includeDescendants = TRUE
)
codes2 <- getCandidateCodes(
  cdm = cdm,
  keywords = c("knee osteoarthritis", "arthrosis"),
  domains = "Condition",
  includeDescendants = TRUE
)
compareCodelists(
  codelist1 = codes1,
  codelist2 = codes2
)
CDMConnector::cdmDisconnect(cdm)
## End(Not run)
```
doseFormToRoute  

Equivalence from dose from concept IDs to route categories.

Description
Equivalence from dose from concept IDs to route categories.

Usage
doseFormToRoute

Format
A data frame with two variables: dose_form_concept_id and route_category.

getATCCodes  

Get descendant codes for ATC levels

Description
Get descendant codes for ATC levels

Usage
getATCCodes(cdm,  
level = c("ATC 1st"),  
name = NULL,  
doseForm = NULL,  
doseUnit = NULL,  
routeCategory = NULL,  
type = "codelist"
)

Arguments
cdm  
cdm_reference via CDMConnector
level  
ATC level. Can be one or more of "ATC 1st", "ATC 2nd", "ATC 3rd", "ATC 4th", and "ATC 5th"
name  
ATC name of interest. For example, c("Dermatologicals", "Nervous System"), would result in a list of length two with the descendant concepts for these two particular ATC groups.
doseForm  
Only descendants codes with the specified dose form will be returned. If NULL, descendant codes will be returned regardless of dose form.
getCandidateCodes

- **doseUnit**: Only descendants codes with the specified dose unit will be returned. If NULL, descendant codes will be returned regardless of dose unit.
- **routeCategory**: Only descendants codes with the specified route will be returned. If NULL, descendant codes will be returned regardless of dose form.
- **type**: Can be "codelist", "codelist_with_details", or "concept_set_expression".

**Value**

Concepts with their format based on the type argument.

**Examples**

```r
## Not run:
cdm <- mockVocabRef()
getATCCodes(cdm = cdm, level = "ATC 1st")
CDMConnector::cdmDisconnect(cdm)
## End(Not run)
```

---

**getCandidateCodes**

Generate candidate codelist for the OMOP CDM

**Description**

This function generates a set of codes that can be considered for creating a phenotype using the OMOP CDM.

**Usage**

```r
getCandidateCodes(
  cdm,
  keywords,
  exclude = NULL,
  domains = "Condition",
  standardConcept = "Standard",
  searchInSynonyms = FALSE,
  searchNonStandard = FALSE,
  includeDescendants = FALSE,
  includeAncestor = FALSE
)
```

**Arguments**

- **cdm**: cdm_reference via CDMConnector
- **keywords**: Character vector of words to search for. Where more than one word is given (e.g. "knee osteoarthritis"), all combinations of those words should be identified positions (e.g. "osteoarthritis of knee") should be identified.
getConceptClassId

**Description**

getConceptClassId

**Usage**

getConceptClassId(cdm, standardConcept = "Standard", domain = NULL)
getDescendants

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdm</td>
<td>cdm_reference via CDMConnector</td>
</tr>
<tr>
<td>conceptId</td>
<td>concept_id to search</td>
</tr>
<tr>
<td>withAncestor</td>
<td>If TRUE, return column with ancestor. In case of multiple ancestors, concepts will be separated by &quot;;&quot;</td>
</tr>
</tbody>
</table>

Value

The concept class used for a given set of domains

Examples

```r
## Not run:
cdm <- mockVocabRef()
getConceptClassId(cdm = cdm, domain = "drug")
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

Description

getDescendants

Usage

```r
getDescendants(
cdm,  
conceptId,  
withAncestor = FALSE,  
ingredientRange = c(0, Inf),  
doseForm = NULL  
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdm</td>
<td>cdm_reference via CDMConnector</td>
</tr>
<tr>
<td>conceptId</td>
<td>concept_id to search</td>
</tr>
<tr>
<td>withAncestor</td>
<td>If TRUE, return column with ancestor. In case of multiple ancestors, concepts will be separated by &quot;;&quot;</td>
</tr>
</tbody>
</table>
getDomains

ingredientRange
Used to restrict descendant codes to those associated with a specific number of drug ingredients. Must be a vector of length two with the first element the minimum number of ingredients allowed and the second the maximum. A value of c(2, 2) would restrict to only concepts associated with two ingredients.

doseForm
Only descendants codes with the specified drug dose form will be returned. If NULL, descendant codes will be returned regardless of dose form.

Value
The descendants of a given concept id

Examples

## Not run:
cdm <- mockVocabRef()
getDescendants(cdm = cdm, conceptId = 1)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)

getDomains
getDomains

Description
getDomains

Usage
getDomains(cdm, standardConcept = "Standard")

Arguments
cdm cdm_reference via CDMConnector
standardConcept Character vector with one or more of "Standard", "Classification", and "Non-standard". These correspond to the flags used for the standard_concept field in the concept table of the cdm.

Value
The domains of the cdm
Description
getDoseForm

Usage
getDoseForm(cdm)

Arguments

  cdm                 cdm_reference via CDMConnector

Value
The dose forms available for drug concepts

Examples

## Not run:
cdm <- mockVocabRef()
getDomains(cdm = cdm)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)

getDoseUnit

Get available routes in a cdm reference.

Description
Get the dose form categories available in the database (see https://doi.org/10.1002/pds.5809) for more details on how routes were classified.

Usage
getDoseUnit(cdm)
Arguments
cdm A cdm reference.

Value
A character vector with available routes

getDrugIngredientCodes
Get descendant codes for drug ingredients

Description
Get descendant codes for drug ingredients

Usage
getDrugIngredientCodes(
  cdm,
  name = NULL,
  doseForm = NULL,
  doseUnit = NULL,
  routeCategory = NULL,
  ingredientRange = c(1, Inf),
  type = "codelist"
)

Arguments
cdm cdm_reference via CDMConnector
name Names of ingredients of interest. For example, c("acetaminophen", "codeine"),
would result in a list of length two with the descendant concepts for these two
particular drug ingredients.
doseForm Only descendants codes with the specified dose form will be returned. If NULL,
descendant codes will be returned regardless of dose form.
doseUnit Only descendants codes with the specified dose unit will be returned. If NULL,
descendant codes will be returned regardless of dose unit
routeCategory Only descendants codes with the specified route will be returned. If NULL,
descendant codes will be returned regardless of route category.
ingredientRange Used to restrict descendant codes to those associated with a specific number of
ingredients. Must be a vector of length two with the first element the minimum
number of ingredients allowed and the second the maximum. A value of c(2, 2)
would restrict to only concepts associated with two ingredients.
type Can be "codelist", "codelist_with_details", or "concept_set_expression"
getICD10StandardCodes

Value

Concepts with their format based on the type argument.

Examples

```r
## Not run:
cdm <- mockVocabRef()
getDrugIngredientCodes(cdm = cdm, name = "Adalimumab")
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

getICD10StandardCodes

Get corresponding standard codes for ICD-10 chapters and sub-chapters

Description

Get corresponding standard codes for ICD-10 chapters and sub-chapters

Usage

```r
getICD10StandardCodes(
  cdm,
  level = c("ICD10 Chapter", "ICD10 SubChapter"),
  name = NULL,
  includeDescendants = TRUE,
  type = "codelist"
)
```

Arguments

- `cdm` - cdm_reference via CDMConnector
- `level` - Can be either "ICD10 Chapter" or "ICD10 SubChapter"
- `name` - Name of chapter or sub-chapter of interest. If NULL, all will be considered.
- `includeDescendants` - If FALSE only direct mappings from ICD-10 codes to standard codes will be returned. If TRUE descendants of standard concepts will also be included.
- `type` - Can be "codelist", "codelist_with_details", or "concept_set_expression"

Value

A named list, with each element containing the corresponding standard codes (and descendants) of ICD chapters and sub-chapters
getMappings

Show mappings from non-standard vocabularies to standard

Description

Show mappings from non-standard vocabularies to standard

Usage

getMappings(
  candidateCodelist,
  cdm = NULL,
  nonStandardVocabularies = c("ATC", "ICD10CM", "ICD10PCS", "ICD9CM", "ICD9Proc", "LOINC", "OPCS4", "Read", "RxNorm", "RxNorm Extension", "SNOMED")
)

Arguments

candidateCodelist
  Dataframe
cdm
  cdm_reference via CDMConnector::cdm_from_con()
nonStandardVocabularies
  Character vector

Value

tibble

Examples

## Not run:

cdm <- mockVocabRef()
getICD10StandardCodes(cdm = cdm, level = c("ICD10 Chapter", "ICD10 SubChapter"))
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
getRelationshipId

```r
cdm = cdm,
candidateCodelist = codes,
nonStandardVocabularies = "READ"
)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

---

**getRelationshipId**  
*Get relationship ID values from the concept relationship table*

---

**Description**

Get relationship ID values from the concept relationship table

**Usage**

```r
getRelationshipId(
  cdm,  
  standardConcept1 = "standard",  
  standardConcept2 = "standard",  
  domains1 = "condition",  
  domains2 = "condition"
)
```

**Arguments**

- **cdm**  
  A cdm reference

- **standardConcept1**  
  Character vector with one or more of "Standard", "Classification", and "Non-standard". These correspond to the flags used for the standard_concept field in the concept table of the cdm.

- **standardConcept2**  
  Character vector with one or more of "Standard", "Classification", and "Non-standard". These correspond to the flags used for the standard_concept field in the concept table of the cdm.

- **domains1**  
  Character vector with one or more of the OMOP CDM domain.

- **domains2**  
  Character vector with one or more of the OMOP CDM domain.

**Value**

A character vector with unique values
getVocabularies

Description
getVocabularies

Usage
getVocabularies(cdm)

Arguments

  cdm               cdm_reference via CDMConnector

Value

  Names of available vocabularies

getRouteCategories

Description
Get available routes in a cdm reference.

Usage
getRouteCategories(cdm)

Arguments

  cdm               A cdm reference.

Value

  A character vector with available routes

Examples

## Not run:
cdm <- mockVocabRef()
getRelationshipId(cdm = cdm)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
getVocabVersion

Examples

## Not run:
cdm <- mockVocabRef()
getVocabularies(cdm = cdm)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)

getVocabVersion

Description

getVocabVersion

Usage

getVocabVersion(cdm)

Arguments

cdm
cdm_reference via CDMConnector

Value

the vocabulary version being used

Examples

## Not run:
cdm <- mockVocabRef()
getVocabVersion(cdm = cdm)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)

mockVocabRef

Generate example vocabulary database

Description

Generate example vocabulary database

Usage

mockVocabRef(backend = "data_frame")
**restrictToCodesInUse**

### Arguments

- **backend**
  - `database` (duckdb) or `data_frame`

### Value

- cdm reference with mock vocabulary

### Examples

```r
## Not run:
cdm <- mockVocabRef()
cdm
CDMConnector::cdmDisconnect(cdm)
## End(Not run)
```

---

**restrictToCodesInUse**

*Use achilles counts to filter a codelist to keep only the codes used in the database*

### Description

Use achilles counts to filter a codelist to keep only the codes used in the database

### Usage

```r
restrictToCodesInUse(  
  x,  
  cdm,  
  minimumCount = 0L,  
  table = c("condition_occurrence", "device_exposure", "drug_exposure", "measurement", "observation", "procedure_occurrence", "visit_occurrence")  
)
```

### Arguments

- **x**
  - A codelist
- **cdm**
  - cdm_reference via CDMConnector
- **minimumCount**
  - Any codes with a frequency under this will be removed.
- **table**
  - cdm table

### Value

Use achilles counts to filter codelist to only the codes used in the database
sourceCodesInUse

Examples

```r
## Not run:
cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
    keywords = "arthritis",
    domains = "Condition",
    includeDescendants = FALSE)
x <- restrictToCodesInUse(list("cs1" = codes$concept_id,
    "cs2" = 999),
    cdm = cdm)

x

CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

sourceCodesInUse  Use achilles counts to get source codes used in the database

Description

Use achilles counts to get source codes used in the database

Usage

```r
sourceCodesInUse(
    cdm,
    table = c("condition_occurrence", "device_exposure", "drug_exposure", "measurement",
              "observation", "procedure_occurrence", "visit_occurrence")
)
```

Arguments

- `cdm`  cdm_reference via CDMConnector
- `table`  cdm table

Value

A list of source codes used in the database.

Examples

```r
## Not run:
cdm <- mockVocabRef("database")
x <- sourceCodesInUse(cdm = cdm)
x

CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```
**stratifyByConcept**  
*Stratify a codelist by the concepts included within it*

**Description**
Stratify a codelist by the concepts included within it

**Usage**

```r
stratifyByConcept(x, cdm, keepOriginal = FALSE)
```

**Arguments**
- **x**: A codelist
- **cdm**: A cdm reference
- **keepOriginal**: Whether to keep the original codelist and append the stratify (if TRUE) or just return the stratified codelist (if FALSE).

**Value**
A codelist

---

**stratifyByDoseUnit**  
*Stratify a codelist by dose unit*

**Description**
Stratify a codelist by dose unit

**Usage**

```r
stratifyByDoseUnit(x, cdm, keepOriginal = FALSE)
```

**Arguments**
- **x**: A codelist
- **cdm**: A cdm reference
- **keepOriginal**: Whether to keep the original codelist and append the stratify (if TRUE) or just return the stratified codelist (if FALSE).

**Value**
A codelist
stratifyByRouteCategory

**Stratify a codelist by route category**

**Description**
Stratify a codelist by route category

**Usage**
```
stratifyByRouteCategory(x, cdm, keepOriginal = FALSE)
```

**Arguments**
- `x`: A codelist
- `cdm`: A cdm reference
- `keepOriginal`: Whether to keep the original codelist and append the stratify (if TRUE) or just return the stratified codelist (if FALSE).

**Value**
A codelist

---

subsetOnDoseUnit

**Subset a codelist to only those with a particular dose unit**

**Description**
Subset a codelist to only those with a particular dose unit

**Usage**
```
subsetOnDoseUnit(x, cdm, doseUnit)
```

**Arguments**
- `x`: Codelist
- `cdm`: A cdm reference
- `doseUnit`: Dose unit. Use getDoseUnit() to find the available dose units in a cdm

**Value**
The codelist with only those concepts associated with the dose unit
**subsetOnRouteCategory**  
*Subset a codelist to only those with a particular route category*

**Description**

Subset a codelist to only those with a particular route category

**Usage**

```r
subsetOnRouteCategory(x, cdm, routeCategory)
```

**Arguments**

- `x`: Codelist
- `cdm`: A cdm reference
- `routeCategory`: Route category. Use `getRoutes()` to find the available route categories for a cdm

**Value**

The codelist with only those concepts associated with the specified route categories

---

**subsetToCodesInUse**  
*Use achilles counts to filter a codelist to keep only the codes used in the database*

**Description**

Use achilles counts to filter a codelist to keep only the codes used in the database

**Usage**

```r
subsetToCodesInUse(
  x, 
  cdm, 
  minimumCount = 0L, 
  table = c("condition_occurrence", "device_exposure", "drug_exposure", "measurement", 
              "observation", "procedure_occurrence", "visit_occurrence")
)
```

**Arguments**

- `x`: A codelist
- `cdm`: cdm_reference via CDMConnector
- `minimumCount`: Any codes with a frequency under this will be removed.
- `table`: cdm table
summariseAchillesCodeUse

Value

Use achilles counts to filter codelist to only the codes used in the database

Examples

```r
## Not run:
cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
    keywords = "arthritis",
    domains = "Condition",
    includeDescendants = FALSE)
x <- subsetToCodesInUse(list("cs1" = codes$concept_id,
    "cs2" = 999),
    cdm = cdm)

x
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

summariseAchillesCodeUse

Summarise code use from achilles counts

Description

Summarise code use from achilles counts

Usage

```r
summariseAchillesCodeUse(
    x,
    cdm,
    countBy = c("record", "person"),
    minCellCount = lifecycle::deprecated()
)
```

Arguments

- `x` Codelist
- `cdm` cdm_reference via CDMConnector::cdm_from_con()
- `countBy` Either "record" for record-level counts or "person" for person-level counts
- `minCellCount` 

Value

A tibble with results
summariseCodeUse

Examples

```r
## Not run:
cdm <- mockVocabRef("database")
oa <- getCandidateCodes(cdm = cdm, keywords = "osteoarthritis")
result_achilles <- summariseAchillesCodeUse(list(oa = oa$concept_id), cdm = cdm)
result_achilles
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

summariseCodeUse  Summarise code use in patient-level data

Description

Summarise code use in patient-level data

Usage

```r
summariseCodeUse(
  x,
  cdm,
  countBy = c("record", "person"),
  byConcept = TRUE,
  byYear = FALSE,
  bySex = FALSE,
  ageGroup = NULL,
  minCellCount = lifecycle::deprecated()
)
```

Arguments

- **x**: List of concept IDs
- **cdm**: cdm_reference via CDMConnector::cdm_from_con()
- **countBy**: Either "record" for record-level counts or "person" for person-level counts
- **byConcept**: TRUE or FALSE. If TRUE code use will be summarised by
- **byYear**: TRUE or FALSE. If TRUE code use will be summarised by year.
- **bySex**: TRUE or FALSE. If TRUE code use will be summarised by sex.
- **ageGroup**: If not NULL, a list of ageGroup vectors of length two.
- **minCellCount**:

Value

A tibble with results overall and, if specified, by strata
summariseCohortCodeUse

Examples

```r
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
  dbdir = CDMConnector::eunomia_dir())
cdm <- CDMConnector::cdm_from_con(con,
  cdm_schem = "main",
  write_schema = "main")
acetiminophen <- c(1125315, 1127433, 40229134,
  40231925, 40162522, 19133768, 1127078)
poliovirus_vaccine <- c(40213160)
cs <- list(acetiminophen = acetiminophen,
  poliovirus_vaccine = poliovirus_vaccine)
results <- summariseCodeUse(cs, cdm = cdm)
results
CDMConnector::cdmDisconnect(cdm)
```

## End(Not run)

summariseCohortCodeUse

*Summarise code use among a cohort in the cdm reference*

Description

Summarise code use among a cohort in the cdm reference

Usage

```r
summariseCohortCodeUse(
  x,
  cdm,  
  cohortTable, 
  cohortId = NULL, 
  timing = "any", 
  countBy = c("record", "person"), 
  byConcept = TRUE, 
  byYear = FALSE, 
  bySex = FALSE, 
  ageGroup = NULL, 
  minCellCount = lifecycle::deprecated()
)
```

Arguments

- `x` Vector of concept IDs
- `cdm` cdm_reference via CDMConnector::cdm_from_con()
cohortTable  A cohort table from the cdm reference.
cohortId  A vector of cohort IDs to include
timing  When to assess the code use relative cohort dates. This can be "any" (code use any time by individuals in the cohort) or "entry" (code use on individuals’ cohort start date).
countBy  Either "record" for record-level counts or "person" for person-level counts
byConcept  TRUE or FALSE. If TRUE code use will be summarised by
byYear  TRUE or FALSE. If TRUE code use will be summarised by year.
bySex  TRUE or FALSE. If TRUE code use will be summarised by sex.
ageGroup  If not NULL, a list of ageGroup vectors of length two.
minCellCount  A tibble with results overall and, if specified, by strata

Value
A tibble with results overall and, if specified, by strata

Examples

## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
dbdir = CDMConnector::eunomia_dir())
cdm <- CDMConnector::cdm_from_con(con,
  cdm_schem = "main",
  write_schema = "main")
cdm <- CDMConnector::generateConceptCohortSet(cdm = cdm,
  conceptSet = list(a = 260139,
    b = 1127433),
  name = "cohorts",
  end = "observation_period_end_date",
  overwrite = TRUE)

results_cohort_mult <-
summariseCohortCodeUse(list(cs = c(260139,19133873)),
  cdm = cdm,
  cohortTable = "cohorts",
  timing = "entry")

results_cohort_mult
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
summariseOrphanCodes

Find orphan codes related to a codelist using achilles counts and, if available, PHOEBE concept recommendations

Description
Find orphan codes related to a codelist using achilles counts and, if available, PHOEBE concept recommendations

Usage
summariseOrphanCodes(
  x,
  cdm,
  domain = c("condition", "device", "drug", "measurement", "observation", "procedure", "visit")
)

Arguments
x A codelist for which to find related codes used in the database
cdm cdm_reference via CDMConnector
domain The domains to restrict results too. Only concepts from these domains will be returned.

Value
A summarised result containing the frequency of codes related to (but not in) the codelist

Examples
## Not run:
cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
  keywords = "Musculoskeletal disorder",
  domains = "Condition",
  includeDescendants = FALSE)

orphan_codes <- summariseOrphanCodes(x = list("msk" = codes$concept_id),
cdm = cdm)

orphan_codes
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
Format the result of summariseAchillesCodeUse into a table.

### Description
Format the result of summariseAchillesCodeUse into a table.

### Usage
```r
tableAchillesCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate"),
  conceptId = TRUE,
  standard = TRUE,
  vocabulary = TRUE,
  groupColumns = NULL,
  excludeColumns = c("result_id", "estimate_type"),
  .options = list()
)
```

### Arguments
- **result**: A summarised result with results of the type "achilles_code_use".
- **type**: Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
- **header**: A vector containing which elements should go into the header in order. Allowed are: cdm_name, group, strata, additional, variable, estimate, settings.
- **conceptId**: If TRUE concept ids will be displayed.
- **standard**: If TRUE a column indicating if the code is standard will be displayed.
- **vocabulary**: If TRUE vocabulary id will be displayed.
- **groupColumns**: Columns to use as group labels. Allowed columns are cdm_name and/or codelist_name.
- **excludeColumns**: Columns to drop from the output table.
- **.options**: Named list with additional formatting options. visOmopResults::optionsVisOmopTable() shows allowed arguments and their default values.

### Value
A table with a formatted version of the summariseCohortCodeUse result.

### Examples
```r
## Not run:
cdm <- mockVocabRef("database")
oa <- get CandidateCodes(cdm = cdm, keywords = "osteoarthritis")
result_achilles <- summariseAchillesCodeUse(list(oa = oa$concept_id), cdm = cdm)
```
tableCodeUse(result_achilles)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)

tableCodeUse Format the result of summariseCodeUse into a table.

Description

Format the result of summariseCodeUse into a table.

Usage

tableCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate"),
  splitStrata = TRUE,
  conceptId = TRUE,
  sourceConcept = TRUE,
  groupColumns = NULL,
  excludeColumns = c("result_id", "estimate_type", "additional_name", "additional_level"),
  .options = list()
)

Arguments

result A summarised result with results of the type "code_use".
type Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
header A vector containing which elements should go into the header in order. Allowed are: cdm_name, group, strata, additional, variable, estimate, settings.
splitStrata If TRUE strata will be split.
conceptId If TRUE concept ids will be displayed.
sourceConcept If TRUE source concepts will be displayed.
groupColumns Columns to use as group labels. Allowed columns are cdm_name and/or codelist_name.
excludeColumns Columns to drop from the output table.
.options Named list with additional formatting options. visOmopResults::optionsVisOmopTable() shows allowed arguments and their default values.

Value

A table with a formatted version of the summariseCodeUse result.
Examples

```r
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
  dbdir = CDMConnector::eunomia_dir())
cdm <- CDMConnector::cdm_from_con(con,
  cdm_schem = "main",
  write_schema = "main")
acetaminophen <- c(1125315, 1127433, 40229134,
  40231925, 40162522, 19133768, 1127078)
poliovirus_vaccine <- c(40213160)
cs <- list(acetaminophen = acetaminophen,
  poliovirus_vaccine = poliovirus_vaccine)
results <- summariseCodeUse(cs, cdm = cdm)
tableCohortCodeUse(results)
CDMConnector::cdmDisconnect(cdm)
## End(Not run)
```

---

tableCohortCodeUse

Format the result of summariseCohortCodeUse into a table.

Description

Format the result of summariseCohortCodeUse into a table.

Usage

```r
tableCohortCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate"),
  splitStrata = TRUE,
  conceptId = TRUE,
  sourceConcept = TRUE,
  timing = FALSE,
  groupColumns = NULL,
  excludeColumns = c("result_id", "estimate_type", "additional_name", "additional_level"),
  .options = list()
)
```

Arguments

- **result**: A summarised result with results of the type "cohort_code_use".
- **type**: Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
- **header**: A vector containing which elements should go into the header in order. Allowed are: cdm_name, group, strata, additional, variable, estimate, settings.
splitStrata  If TRUE strata will be split.
conceptId   If TRUE concept ids will be displayed.
sourceConcept If TRUE source concepts will be displayed.
timing      If TRUE the timing setting will be displayed.
groupColumns Columns to use as group labels. Allowed columns are cdm_name, cohort_name and/or codelist_name.
excludeColumns Columns to drop from the output table.
.options     Named list with additional formatting options. visOmopResults::optionsVisOmopTable() shows allowed arguments and their default values.

Value
A table with a formatted version of the summariseCohortCodeUse result.

Examples
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
  dbdir = CDMConnector::eunomia_dir())
cdm <- CDMConnector::cdm_from_con(con,
  cdm_schem = "main",
  write_schema = "main")
cdm <- CDMConnector::generateConceptCohortSet(cdm = cdm,
  conceptSet = list(a = 260139,
    b = 1127433),
  name = "cohorts",
  end = "observation_period_end_date",
  overwrite = TRUE)

results_cohort_mult <-
summariseCohortCodeUse(list(cs = c(260139, 19133873)),
  cdm = cdm,
  cohortTable = "cohorts",
  timing = "entry")

tableCohortCodeUse(results_cohort_mult)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
Usage

tableOrphanCodes(
  result,
  type = "gt",
  header = c("cdm_name", "estimate"),
  conceptId = TRUE,
  standard = TRUE,
  vocabulary = TRUE,
  groupColumns = NULL,
  excludeColumns = c("result_id", "estimate_type"),
  .options = list()
)

Arguments

result A summarised result with results of the type "orphan_codes".

type Type of desired formatted table, possibilities: "gt", "flextable", "tibble".

header A vector containing which elements should go into the header in order. Allowed are: cdm_name, group, strata, additional, variable, estimate, settings.

conceptId If TRUE concept ids will be displayed.

standard If TRUE a column indicating if the code is standard will be displayed.

vocabulary If TRUE vocabulary id will be displayed.

groupColumns Columns to use as group labels. Allowed columns are cdm_name and/or codelist_name.

excludeColumns Columns to drop from the output table.

.options Named list with additional formatting options. visOmopResults::optionsVisOmopTable() shows allowed arguments and their default values.

Value

A table with a formatted version of the summariseOrphanCodes result.

Examples

```r
## Not run:
cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
  keywords = "Musculoskeletal disorder",
  domains = "Condition",
  includeDescendants = FALSE)

orphan_codes <- summariseOrphanCodes(x = list("msk" = codes$concept_id),
  cdm = cdm,
  domains = "Condition",
  standardConcept = "Standard",
  searchInSynonyms = FALSE,
  searchNonStandard = FALSE,
  includeDescendants = TRUE,
  .options = list()
```
includeAncestor = FALSE)
tableOrphanCodes(orphan_codes)

CDMConnector::cdmDisconnect(cdm)

## End(Not run)
Index

* datasets
  
doseFormToRoute, 6
d codesFromCohort, 3
codesFromConceptSet, 3
codesInUse, 4
curcompareCodelists, 5
doseFormToRoute, 6

g etATCCodes, 6
g etCandidateCodes, 7
g etConceptClassId, 8
g etDescendants, 9
g etDomains, 10
g etDoseForm, 11
g etDoseUnit, 11
g etDrugIngredientCodes, 12
g etICD10StandardCodes, 13
g etMappings, 14
g etRelationshipId, 15
g etRouteCategories, 16
g etVocabularies, 16
g etVocabVersion, 17

mockVocabRef, 17

restrictToCodesInUse, 18

sourceCodesInUse, 19
stratifyByConcept, 20
stratifyByDoseUnit, 20
stratifyByRouteCategory, 21
subsetOnDoseUnit, 21
subsetOnRouteCategory, 22
subsetToCodesInUse, 22
summariseAchillesCodeUse, 23
summariseCodeUse, 24
summariseCohortCodeUse, 25
summariseOrphanCodes, 27
t ableAchillesCodeUse, 28
t ableCodeUse, 29
t ableCohortCodeUse, 30
t ableOrphanCodes, 31