Package 'CodelistGenerator'

April 10, 2025

Title Identify Relevant Clinical Codes and Evaluate Their Use

Version 3.5.0

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 (>= 4.1)

Imports DBI (>= 1.1.0), dplyr (>= 1.1.0), omopgenerics (>= 1.1.1), rlang (>= 1.0.0), glue (>= 1.5.0), stringr (>= 1.4.0), stringi (>= 1.8.1), tidyr (>= 1.2.0), cli (>= 3.1.0), purrr, clock, PatientProfiles (>= 1.3.1), vctrs, jsonlite, lifecycle

Suggests covr, duckdb, CDMConnector (>= 1.7.0), visOmopResults (>= 1.0.0), knitr, rmarkdown, testthat (>= 3.0.0), RPostgres, odbc, spelling, tibble, gt, flextable, omock, CohortConstructor

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

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

Language en-US

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 [aut] (<https://orcid.org/0009-0001-8112-8959>), Nuria Mercade-Besora [aut] (<https://orcid.org/0009-0006-7948-3747>), Mike Du [ctb] (<https://orcid.org/0000-0002-9517-8834>), Danielle Newby [ctb] (<https://orcid.org/0000-0002-3001-1478>), Marta Alcalde-Herraiz [ctb] (<https://orcid.org/0009-0002-4405-1814>)

Maintainer Edward Burn <edward.burn@ndorms.ox.ac.uk>

Repository CRAN

Date/Publication 2025-04-10 19:20:39 UTC

Contents

availableATC *Get the names of all available Anatomical Therapeutic Chemical (ATC) classification codes*

Description

Get the names of all available Anatomical Therapeutic Chemical (ATC) classification codes

Usage

```
availableATC(cdm, level = c("ATC 1st"))
```

Arguments

cdm	A cdm reference via CDMConnector.
level	ATC level. Can be one or more of "ATC 1st", "ATC 2nd", "ATC 3rd", "ATC 4th", and "ATC 5th".

Value

A vector containing the names of ATC codes for the chosen level(s) found in the concept table of cdm.

Examples

```
cdm <- mockVocabRef()
availableATC(cdm)
```

availableICD10 *Get the names of all International Classification of Diseases (ICD) 10 codes*

Description

Get the names of all International Classification of Diseases (ICD) 10 codes

Usage

```
availableICD10(cdm, level = c("ICD10 Chapter", "ICD10 SubChapter"))
```

Arguments

cdm	A cdm reference via CDMConnector.
level	Can be either "ICD10 Chapter", "ICD10 SubChapter", "ICD10 Hierarchy", or "ICD10 Code".

Value

A vector containing the names of all ICD-10 codes for the chosen level(s) found in the concept table of cdm.

Examples

```
cdm <- mockVocabRef()
availableICD10(cdm)
```

`availableIngredients` *Get the names of all available drug ingredients*

Description

Get the names of all available drug ingredients

Usage

```
availableIngredients(cdm)
```

Arguments

`cdm`	A cdm reference via CDMConnector.

Value

A vector containing the concept names for all ingredient level codes found in the concept table of cdm.

Examples

```
cdm <- mockVocabRef()
availableIngredients(cdm)
```

buildAchillesTables *Add the achilles tables with specified analyses*

Description

If the cdm reference does not contain the achilles tables, this function will create them for the analyses used by other functions in the package.

Usage

```
buildAchillesTables(cdm, achillesId = NULL)
```

Arguments

cdm	A cdm reference via CDMConnector.
achillesId	A vector of achilles ids. If NULL default analysis will be used.

Value

The cdm_reference object with the achilles tables populated.

Examples

```
dbName <- "GiBleed"
CDMConnector::requireEunomia(dbName)
con <- duckdb::dbConnect(duckdb::duckdb(), CDMConnector::eunomiaDir(dbName))
cdm <- CDMConnector::cdmFromCon(
  con = con, cdmSchema = "main", writeSchema = "main"
)

cdm <- buildAchillesTables(cdm = cdm)
```

codesFromCohort *Get concept ids from JSON files containing cohort definitions*

Description

Get concept ids from JSON files containing cohort definitions

Usage

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

Arguments

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

Value

Named list with concept_ids for each concept set.

Examples

```
cdm <- mockVocabRef("database")
x <- codesFromCohort(cdm = cdm,
                     path =  system.file(package = "CodelistGenerator",
                     "cohorts_for_mock"))
x
CDMConnector::cdmDisconnect(cdm)
```

---

codesFromConceptSet *Get concept ids from JSON files containing concept sets*

Description

Get concept ids from JSON files containing concept sets

Usage

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

Arguments

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

Value

Named list with concept_ids for each concept set.

Examples

```
cdm <- mockVocabRef("database")
x <- codesFromConceptSet(cdm = cdm,
                         path =  system.file(package = "CodelistGenerator",
                         "concepts_for_mock"))
x
CDMConnector::cdmDisconnect(cdm)
```

`codesInUse` *Get the concepts being used in patient records*

Description

Get the concepts being used in patient records

Usage

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

Arguments

`cdm`	A cdm reference via CDMConnector.
`minimumCount`	Any codes with a frequency under this will be removed.
`table`	cdm table of interest.

Value

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

Examples

```
cdm <- mockVocabRef("database")
x <- codesInUse(cdm = cdm)
x
CDMConnector::cdmDisconnect(cdm)
```

`compareCodelists` *Compare overlap between two sets of codes*

Description

Compare overlap between two sets of codes

Usage

```
compareCodelists(codelist1, codelist2)
```

Arguments

`codelist1`	Output of getCandidateCodes or a codelist
`codelist2`	Output of getCandidateCodes.

Value

Tibble with information on the overlap of codes in both codelists.

Examples

```
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
)
```

getATCCodes *Get the descendant codes of Anatomical Therapeutic Chemical (ATC) classification codes*

Description

Get the descendant codes of Anatomical Therapeutic Chemical (ATC) classification codes

Usage

```
getATCCodes(
  cdm,
  level = c("ATC 1st"),
  name = NULL,
  nameStyle = "{concept_code}_{concept_name}",
  doseForm = NULL,
  doseUnit = NULL,
  routeCategory = NULL,
  type = "codelist"
)
```

Arguments

cdm	A 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.
nameStyle	Name style to apply to returned list. Can be one of "{concept_code}","{concept_id}", "{concept_name}", or a combination (i.e., "{concept_code}_{concept_name}").
doseForm	Only codes with the specified dose form will be returned. If NULL, descendant codes will be returned regardless of dose form. Use 'getDoseForm()' to see the available dose forms.
doseUnit	Only codes with the specified dose unit will be returned. If NULL, descendant codes will be returned regardless of dose unit Use 'getDoseUnit()' to see the available dose units.
routeCategory	Only codes with the specified route will be returned. If NULL, descendant codes will be returned regardless of route category. Use getRoutes() to find the available route categories.
type	Can be "codelist" or "codelist_with_details".

Value

Concepts with their format based on the type argument

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
getATCCodes(cdm = cdm, level = "ATC 1st")
```

`getCandidateCodes` *Generate a candidate codelist*

Description

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

Usage

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

Arguments

`cdm`	A 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.
`exclude`	Character vector of words to identify concepts to exclude.
`domains`	Character vector with one or more of the OMOP CDM domain. If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.
`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.
`searchInSynonyms`	Either TRUE or FALSE. If TRUE the code will also search using both the primary name in the concept table and synonyms from the concept synonym table.

searchNonStandard
: Either TRUE or FALSE. If TRUE the code will also search via non-standard concepts.

includeDescendants
: Either TRUE or FALSE. If TRUE descendant concepts of identified concepts will be included in the candidate codelist. If FALSE only direct mappings from ICD-10 codes to standard codes will be returned.

includeAncestor
: Either TRUE or FALSE. If TRUE the direct ancestor concepts of identified concepts will be included in the candidate codelist.

Value

A tibble with the information on the potential codes of interest.

Examples

```
cdm <- CodelistGenerator::mockVocabRef()
CodelistGenerator::getCandidateCodes(
  cdm = cdm,
  keywords = "osteoarthritis"
 )
```

---

getConceptClassId *Get the concept classes used in a given set of domains*

Description

Get the concept classes used in a given set of domains

Usage

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

Arguments

cdm
: A 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.

domain
: Character vector with one or more of the OMOP CDM domains. The results will be restricted to the given domains. Check the available ones by running getDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.

Value

The concept classes used for the requested domains.

Examples

```
cdm <- mockVocabRef()
getConceptClassId(cdm = cdm, domain = "drug")
```

`getDescendants` *Get descendant codes for a given concept*

Description

Get descendant codes for a given concept

Usage

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

Arguments

`cdm`	A cdm reference via CDMConnector.
`conceptId`	concept_id to search
`withAncestor`	If TRUE, return column with ancestor. In case of multiple ancestors, concepts will be separated by ";".
`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 codes with the specified dose form will be returned. If NULL, descendant codes will be returned regardless of dose form. Use 'getDoseForm()' to see the available dose forms.

Value

The descendants of a given concept id.

Examples

```
cdm <- mockVocabRef()
getDescendants(cdm = cdm, conceptId = 1)
```

`getDomains` *Get the domains available in the cdm*

Description

Get the domains available in the cdm

Usage

```
getDomains(cdm, standardConcept = "Standard")
```

Arguments

`cdm`	A 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

A vector with the domains of the cdm.

Examples

```
cdm <- mockVocabRef()
getDomains(cdm = cdm)
```

`getDoseForm` *Get the dose forms available for drug concepts*

Description

Get the dose forms available for drug concepts

Usage

```
getDoseForm(cdm)
```

Arguments

cdm	A cdm reference via CDMConnector.

Value

The dose forms available for drug concepts.

Examples

```
cdm <- mockVocabRef()
getDoseForm(cdm = cdm)
```

---

getDoseUnit *Get available dose units*

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 via CDMConnector.

Value

A character vector with available routes.

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()

getDoseUnit(cdm)
```

getDrugIngredientCodes *Get descendant codes of drug ingredients*

Description

Get descendant codes of drug ingredients

Usage

```
getDrugIngredientCodes(
  cdm,
  name = NULL,
  nameStyle = "{concept_code}_{concept_name}",
  doseForm = NULL,
  doseUnit = NULL,
  routeCategory = NULL,
  ingredientRange = c(1, Inf),
  type = "codelist"
)
```

Arguments

cdm	A 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. Users can also specify the concept ID instead of the name (e.g., c(1125315, 42948451)) using a numeric vector.
nameStyle	Name style to apply to returned list. Can be one of "{concept_code}","{concept_id}", "{concept_name}", or a combination (i.e., "{concept_code}_{concept_name}").
doseForm	Only codes with the specified dose form will be returned. If NULL, descendant codes will be returned regardless of dose form. Use 'getDoseForm()' to see the available dose forms.
doseUnit	Only codes with the specified dose unit will be returned. If NULL, descendant codes will be returned regardless of dose unit Use 'getDoseUnit()' to see the available dose units.
routeCategory	Only codes with the specified route will be returned. If NULL, descendant codes will be returned regardless of route category. Use getRoutes() to find the available route categories.
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.
type	Can be "codelist" or "codelist_with_details".

Value

Concepts with their format based on the type argument.

Examples

```
cdm <- mockVocabRef()
getDrugIngredientCodes(cdm = cdm, name = "Adalimumab",
                       nameStyle = "{concept_name}")
```

`getICD10StandardCodes` *Get corresponding standard codes for International Classification of Diseases (ICD) 10 codes*

Description

Get corresponding standard codes for International Classification of Diseases (ICD) 10 codes

Usage

```
getICD10StandardCodes(
  cdm,
  level = c("ICD10 Chapter", "ICD10 SubChapter"),
  name = NULL,
  nameStyle = "{concept_code}_{concept_name}",
  includeDescendants = TRUE,
  type = "codelist"
)
```

Arguments

`cdm`	A cdm reference via CDMConnector.
`level`	Can be either "ICD10 Chapter", "ICD10 SubChapter", "ICD10 Hierarchy", or "ICD10 Code".
`name`	Name of chapter or sub-chapter of interest. If NULL, all will be considered.
`nameStyle`	Name style to apply to returned list. Can be one of `"{concept_code}"`,`"{concept_id}"`, `"{concept_name}"`, or a combination (i.e., `"{concept_code}_{concept_name}"`).
`includeDescendants`	Either TRUE or FALSE. If TRUE descendant concepts of identified concepts will be included in the candidate codelist. If FALSE only direct mappings from ICD-10 codes to standard codes will be returned.
`type`	Can be "codelist" or "codelist_with_details".

Value

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

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
getICD10StandardCodes(cdm = cdm, level = c(
  "ICD10 Chapter",
  "ICD10 SubChapter"
))
```

`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`	A cdm reference via CDMConnector.
`nonStandardVocabularies`	Character vector.

Value

Tibble with the information of potential standard to non-standard mappings for the codelist of interest.

Examples

```
cdm <- CodelistGenerator::mockVocabRef()
codes <- CodelistGenerator::getCandidateCodes(
  cdm = cdm,
  keywords = "osteoarthritis"
)
CodelistGenerator::getMappings(
  cdm = cdm,
  candidateCodelist = codes,
```

```
  nonStandardVocabularies = "READ"
)
```

`getRelationshipId` *Get available relationships between concepts*

Description

Get available relationships between concepts

Usage

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

Arguments

`cdm`	A cdm reference via CDMConnector.
`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 concept relationship values.

Examples

```
cdm <- mockVocabRef()
getRelationshipId(cdm = cdm)
```

getRouteCategories *Get available drug routes*

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

```
getRouteCategories(cdm)
```

Arguments

cdm	A cdm reference via CDMConnector.

Value

A character vector with all available routes.

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()

getRouteCategories(cdm)
```

getVocabularies *Get the vocabularies available in the cdm*

Description

Get the vocabularies available in the cdm

Usage

```
getVocabularies(cdm)
```

Arguments

cdm	A cdm reference via CDMConnector.

Value

Names of available vocabularies.

Examples

```
cdm <- mockVocabRef()
getVocabularies(cdm = cdm)
```

---

getVocabVersion *Get the version of the vocabulary used in the cdm*

Description

Get the version of the vocabulary used in the cdm

Usage

```
getVocabVersion(cdm)
```

Arguments

cdm	A cdm reference via CDMConnector.

Value

The vocabulary version being used in the cdm.

Examples

```
cdm <- mockVocabRef()
getVocabVersion(cdm = cdm)
```

---

mockVocabRef *Generate example vocabulary database*

Description

Generate example vocabulary database

Usage

```
mockVocabRef(backend = "data_frame")
```

Arguments

backend	'database' (duckdb) or 'data_frame'.

Value

cdm reference with mock vocabulary.

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
cdm
```

---

sourceCodesInUse *Get the source codes being used in patient records*

Description

Get the source codes being used in patient records

Usage

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

Arguments

cdm	A cdm reference via CDMConnector.
table	cdm table of interest.

Value

A list of source codes used in the database.

Examples

```
cdm <- mockVocabRef("database")
x <- sourceCodesInUse(cdm = cdm)
x
CDMConnector::cdmDisconnect(cdm)
```

stratifyByConcept *Stratify a codelist by the concepts included within it.*

Description

Stratify a codelist by the concepts included within it.

Usage

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

Arguments

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

Value

The codelist or a codelist with details with the required stratifications, as different elements of the list.

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
codes <- list("concepts" = c(20,21))
new_codes <- stratifyByConcept(x = codes,
                               cdm = cdm,
                               keepOriginal = TRUE)
new_codes
```

stratifyByDoseUnit *Stratify a codelist by dose unit.*

Description

Stratify a codelist by dose unit.

Usage

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

Arguments

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

Value

The codelist with the required stratifications, as different elements of the list.

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
codes <- list("concepts" = c(20,21))
new_codes <- stratifyByDoseUnit(x = codes,
                                cdm = cdm,
                                keepOriginal = TRUE)
new_codes
```

---

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 via CDMConnector.
keepOriginal	Whether to keep the original codelist and append the stratify (if TRUE) or just return the stratified codelist (if FALSE).

Value

The codelist with the required stratifications, as different elements of the list.

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
codes <- list("concepts" = c(20,21))
new_codes <- stratifyByRouteCategory(x = codes,
                                     cdm = cdm,
                                     keepOriginal = TRUE)
new_codes
```

`subsetOnDomain` *Subset a codelist to only those codes from a particular domain.*

Description

Subset a codelist to only those codes from a particular domain.

Usage

```
subsetOnDomain(x, cdm, domain, negate = FALSE)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference via CDMConnector.
`domain`	Character vector with one or more of the OMOP CDM domains. The results will be restricted to the given domains. Check the available ones by running getDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.
`negate`	If FALSE, only concepts with the domain specified will be returned. If TRUE, concepts with the domain specified will be excluded.

Value

The codelist with only those concepts associated with the domain (if negate = FALSE) or the codelist without those concepts associated with the domain (if negate = TRUE).

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
codes <- subsetOnDomain(
             x = list("codes" = c(10,13,15)),
             cdm = cdm,
             domain = "Drug")
codes
```

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, negate = FALSE)
```

Arguments

x	A codelist.
cdm	A cdm reference via CDMConnector.
doseUnit	Only codes with the specified dose unit will be returned. If NULL, descendant codes will be returned regardless of dose unit Use 'getDoseUnit()' to see the available dose units.
negate	If FALSE, only concepts with the dose unit specified will be returned. If TRUE, concepts with the dose unit specified will be excluded.

Value

The codelist with only those concepts associated with the dose unit (if negate = FALSE) or codelist without those concepts associated with the dose unit(if negate = TRUE).

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
codes <- subsetOnDoseUnit(x = list("codes" = c(20,21)),
                          cdm = cdm,
                          doseUnit = c("milligram"))

codes
```

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

```
subsetOnRouteCategory(x, cdm, routeCategory, negate = FALSE)
```

Arguments

x	A codelist.
cdm	A cdm reference via CDMConnector.
routeCategory	Only codes with the specified route will be returned. If NULL, descendant codes will be returned regardless of route category. Use getRoutes() to find the available route categories.
negate	If FALSE, only concepts with the routeCategory specified will be returned. If TRUE, concepts with the routeCategory specified will be excluded.

Value

The codelist with only those concepts associated with the specified route categories (if negate is FALSE) or the codelist without those concepts associated with the specified route categories (if negate is TRUE).

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
codes <- subsetOnRouteCategory(
              x = list("codes" = c(20,21)),
              cdm = cdm,
              routeCategory = "topical")
codes
```

---

subsetToCodesInUse *Filter a codelist to keep only the codes being used in patient records*

Description

Filter a codelist to keep only the codes being used in patient records

Usage

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

Arguments

x	A codelist.
cdm	A cdm reference via CDMConnector.
minimumCount	Any codes with a frequency under this will be removed.
table	cdm table of interest.

Value

The filtered codelist with only the codes used in the database

Examples

```
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)
```

---

summariseAchillesCodeUse

Summarise code use from achilles counts.

Description

Summarise code use from achilles counts.

Usage

```
summariseAchillesCodeUse(x, cdm, countBy = c("record", "person"))
```

Arguments

x	A codelist.
cdm	A cdm reference via CDMConnector.
countBy	Either "record" for record-level counts or "person" for person-level counts.

Value

A tibble with summarised counts.

Examples

```
cdm <- mockVocabRef("database")
oa <- getCandidateCodes(cdm = cdm, keywords = "osteoarthritis")
result_achilles <- summariseAchillesCodeUse(list(oa = oa$concept_id), cdm = cdm)
result_achilles
CDMConnector::cdmDisconnect(cdm)
```

`summariseCodeUse` *Summarise code use in patient-level data.*

Description

Summarise code use in patient-level data.

Usage

```
summariseCodeUse(
  x,
  cdm,
  countBy = c("record", "person"),
  byConcept = TRUE,
  byYear = FALSE,
  bySex = FALSE,
  ageGroup = NULL,
  dateRange = as.Date(c(NA, NA))
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference via CDMConnector.
`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 concept.
`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.
`dateRange`	Two dates. The first indicating the earliest cohort start date and the second indicating the latest possible cohort end date. If NULL or the first date is set as missing, the earliest observation_start_date in the observation_period table will be used for the former. If NULL or the second date is set as missing, the latest observation_end_date in the observation_period table will be used for the latter.

Value

A tibble with count results overall and, if specified, by strata.

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
                      dbdir = CDMConnector::eunomiaDir())
cdm <- CDMConnector::cdmFromCon(con,
                                cdmSchema = "main",
                                writeSchema = "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

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

Arguments

`x`	A codelist.
`cdm`	A cdm reference via CDMConnector.
`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 concept.
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.

Value

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

Examples

```
## Not run:
library(CodelistGenerator)
library(duckdb)
library(DBI)
library(CDMConnector)
con <- dbConnect(duckdb(),
                 dbdir = eunomiaDir())
cdm <- cdmFromCon(con,
                 cdmSchema = "main",
                 writeSchema = "main")
cdm <- 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.
cdm	A cdm reference via CDMConnector.
domain	Character vector with one or more of the OMOP CDM domains. The results will be restricted to the given domains. Check the available ones by running getDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.

Value

A summarised result containg the frequency of codes related to (but not in) the codelist.

Examples

```
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)
```

summariseUnmappedCodes *Find unmapped concepts related to codelist*

Description

Find unmapped concepts related to codelist

Usage

```
summariseUnmappedCodes(
  x,
  cdm,
 table = c("condition_occurrence", "device_exposure", "drug_exposure", "measurement",
    "observation", "procedure_occurrence")
)
```

Arguments

x	A codelist.
cdm	A cdm reference via CDMConnector.
table	Names of clinical tables in which to search for unmapped codes. Can be one or more of "condition_occurrence", "device_exposure", "drug_exposure", "measurement", "observation", and "procedure_occurrence".

Value

A summarised result of unmapped concepts related to given codelist.

Examples

```
cdm <- mockVocabRef("database")
codes <- list("Musculoskeletal disorder" = 1)
cdm <- omopgenerics::insertTable(cdm, "condition_occurrence",
dplyr::tibble(person_id = 1,
              condition_occurrence_id = 1,
              condition_concept_id = 0,
              condition_start_date  = as.Date("2000-01-01"),
              condition_type_concept_id  = NA,
              condition_source_concept_id = 7))
summariseUnmappedCodes(x = list("osteoarthritis" = 2), cdm = cdm,
table = "condition_occurrence")

CDMConnector::cdmDisconnect(cdm)
```

`tableAchillesCodeUse` *Format the result of summariseAchillesCodeUse into a table*

Description

Format the result of summariseAchillesCodeUse into a table

Usage

```
tableAchillesCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumn = character(),
  hide = character(),
  .options = list()
)
```

Arguments

`result`	A <summarised_result> with results of the type "achilles_code_use".
`type`	Type of desired formatted table. To see supported formats use visOmopResults::tableType().
`header`	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The header vector can contain one of the following variables: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". Alternatively, it can include other names to use as overall header labels.
`groupColumn`	Variables to use as group labels. Allowed columns are: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". These cannot be used in header.
`hide`	Table columns to exclude, options are: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". These cannot be used in header or groupColumn.
`.options`	Named list with additional formatting options. visOmopResults::tableOptions() shows allowed arguments and their default values.

Value

A table with a formatted version of the summariseCohortCodeUse result.

Examples

```
cdm <- mockVocabRef("database")
oa <- getCandidateCodes(cdm = cdm, keywords = "osteoarthritis")
result_achilles <- summariseAchillesCodeUse(list(oa = oa$concept_id), cdm = cdm)
tableAchillesCodeUse(result_achilles)
CDMConnector::cdmDisconnect(cdm)
```

---

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_name"),
  groupColumn = character(),
  hide = character(),
  .options = list()
)
```

Arguments

result	A <summarised_result> with results of the type "code_use".
type	Type of desired formatted table. To see supported formats use visOmopResults::tableType().
header	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The header vector can contain one of the following variables: "cdm_name", "codelist_name", "standard_concept_name", "standard_concept_id", "estimate_name", "source_concept_name", "source_concept_id", "domain_id". If results are stratified, "year", "sex", "age_group" can also be used. Alternatively, it can include other names to use as overall header labels.
groupColumn	Variables to use as group labels. Allowed columns are: "cdm_name", "codelist_name", "standard_concept_name", "standard_concept_id", "estimate_name", "source_concept_name", "source_concept_id", "domain_id". If results are stratified, "year", "sex", "age_group" can also be used. These cannot be used in header.

hide	Table columns to exclude, options are: "cdm_name", "codelist_name", "year", "sex", "age_group", "standard_concept_name", "standard_concept_id", "estimate_name", "source_concept_name", "source_concept_id", "domain_id". If results are stratified, "year", "sex", "age_group" can also be used. These cannot be used in header or groupColumn.
.options	Named list with additional formatting options. visOmopResults::tableOptions() shows allowed arguments and their default values.

Value

A table with a formatted version of the summariseCodeUse result.

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
                      dbdir = CDMConnector::eunomiaDir())
cdm <- CDMConnector::cdmFromCon(con,
                                cdmSchema = "main",
                                writeSchema = "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)
tableCodeUse(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

```
tableCohortCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumn = character(),
  hide = c("timing"),
  .options = list(),
```

```
  timing = lifecycle::deprecated()
)
```

Arguments

result	A <summarised_result> with results of the type "cohort_code_use".
type	Type of desired formatted table. To see supported formats use visOmopResults::tableType().
header	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The header vector can contain one of the following variables: "cdm_name", "codelist_name", "standard_concept_name", "standard_concept_id", "estimate_name", "source_concept_name", "source_concept_id", "domain_id". If results are stratified, "year", "sex", "age_group" can also be used. Alternatively, it can include other names to use as overall header labels.
groupColumn	Variables to use as group labels. Allowed columns are: "cdm_name", "codelist_name", "standard_concept_name", "standard_concept_id", "estimate_name", "source_concept_name", "source_concept_id", "domain_id". If results are stratified, "year", "sex", "age_group" can also be used. These cannot be used in header.
hide	Table columns to exclude, options are: "cdm_name", "codelist_name", "year", "sex", "age_group", "standard_concept_name", "standard_concept_id", "estimate_name", "source_concept_name", "source_concept_id", "domain_id". If results are stratified, "year", "sex", "age_group" can also be used. These cannot be used in header or groupColumn.
.options	Named list with additional formatting options. visOmopResults::tableOptions() shows allowed arguments and their default values.
timing	deprecated.

Value

A table with a formatted version of the summariseCohortCodeUse result.

Examples

```
## Not run:
con <- DBI::dbConnect(duckdb::duckdb(),
                      dbdir = CDMConnector::eunomiaDir())
cdm <- CDMConnector::cdmFromCon(con,
                                cdmSchema = "main",
                                writeSchema = "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)
```

`tableOrphanCodes` *Format the result of summariseOrphanCodes into a table*

Description

Format the result of summariseOrphanCodes into a table

Usage

```
tableOrphanCodes(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumn = character(),
  hide = character(),
  .options = list()
)
```

Arguments

`result`	A <summarised_result> with results of the type "orphan_codes".
`type`	Type of desired formatted table. To see supported formats use visOmopResults::tableType().
`header`	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The header vector can contain one of the following variables: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". Alternatively, it can include other names to use as overall header labels.
`groupColumn`	Variables to use as group labels. Allowed columns are: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". These cannot be used in header.
`hide`	Table columns to exclude, options are: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". These cannot be used in header or groupColumn.
`.options`	Named list with additional formatting options. visOmopResults::tableOptions() shows allowed arguments and their default values.

Value

A table with a formatted version of the summariseOrphanCodes result.

Examples

```
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)

tableOrphanCodes(orphan_codes)

CDMConnector::cdmDisconnect(cdm)
```

---

tableUnmappedCodes *Format the result of summariseUnmappedCodeUse into a table*

Description

Format the result of summariseUnmappedCodeUse into a table

Usage

```
tableUnmappedCodes(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumn = character(),
  hide = character(),
  .options = list()
)
```

Arguments

result	A <summarised_result> with results of the type "umapped_codes".
type	Type of desired formatted table. To see supported formats use visOmopResults::tableType().
header	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The header vector can contain one of the following variables: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". Alternatively, it can include other names to use as overall header labels.

groupColumn	Variables to use as group labels. Allowed columns are: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". These cannot be used in header.
hide	Table columns to exclude, options are: "cdm_name", "codelist_name", "domain_id", "standard_concept_name", "standard_concept_id", "estimate_name", "standard_concept", "vocabulary_id". These cannot be used in header or groupColumn.
.options	Named list with additional formatting options. visOmopResults::tableOptions() shows allowed arguments and their default values.

Value

A table with a formatted version of the summariseUnmappedCodes result.

Examples

```
cdm <- mockVocabRef("database")
codes <- list("Musculoskeletal disorder" = 1)
cdm <- omopgenerics::insertTable(cdm, "condition_occurrence",
dplyr::tibble(person_id = 1,
              condition_occurrence_id = 1,
              condition_concept_id = 0,
              condition_start_date  = as.Date("2000-01-01"),
              condition_type_concept_id  = NA,
              condition_source_concept_id = 7))
unmapped_codes <- summariseUnmappedCodes(x = list("osteoarthritis" = 2),
cdm = cdm, table = "condition_occurrence")
tableUnmappedCodes(unmapped_codes)

cdm <- omopgenerics::insertTable(
 cdm,
 "measurement",
 dplyr::tibble(
   person_id = 1,
   measurement_id = 1,
   measurement_concept_id = 0,
   measurement_date  = as.Date("2000-01-01"),
   measurement_type_concept_id  = NA,
   measurement_source_concept_id = 7
 )
)
table <- summariseUnmappedCodes(x = list("cs" = 2),
                                cdm = cdm,
                                table = c("measurement"))
tableUnmappedCodes(unmapped_codes)

CDMConnector::cdmDisconnect(cdm)
```

Index