Package 'CodelistGenerator'

May 31, 2024

Title Identify Relevant Clinical Codes and Evaluate Their Use

Version 3.0

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 7.3.1

Depends R (>= 3.5.0)

Imports checkmate (>= 2.0.0), DBI (>= 1.1.0), dplyr (>= 1.1.0), magrittr (>= 2.0.0), omopgenerics (>= 0.2.0), 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 (>= 1.0.0), 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

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/0009-0001-8112-8959>)

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

Repository CRAN

Date/Publication 2024-05-31 07:10:10 UTC

R topics documented:

`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, withConceptDetails = FALSE)
```

Arguments

path	Path to a file or folder containing JSONs of cohort definitions
cdm	A cdm reference created with CDMConnector
withConceptDetails	If FALSE a vector of concept IDs will be returned for each concept set. If TRUE a tibble will be returned with additional information on the identified concepts.

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, withConceptDetails = FALSE)
```

Arguments

path	Path to a file or folder containing JSONs of concept sets
cdm	A cdm reference created with CDMConnector
withConceptDetails	If FALSE a vector of concept IDs will be returned for each concept set. If TRUE a tibble will be returned with additional information on the identified concepts.

Value

Named list with concept_ids for each concept set

Examples

```
## 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 *Get codes used in the database*

Description

Get codes used in the database

Usage

```
codesInUse(
  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 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

```
## 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)
```

---

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,
  withConceptDetails = FALSE
)
```

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.
withConceptDetails	If FALSE, each item in the list of results (one per ATC group) will contain a vector of concept IDs for each ingredient. If TRUE each item in the list of results will contain a tibble with additional information on the identified concepts.

Value

A named list, with each item containing a vector of descendant concepts of an ATC group (if withConceptDetails was set as FALSE) or a tibble with the descendant concepts along with additional details about them (if withConceptDetails was set as TRUE).

Examples

```
## 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

```
getCandidateCodes(
  cdm,
  keywords,
  exclude = NULL,
  domains = "Condition",
  standardConcept = "Standard",
  searchInSynonyms = FALSE,
```

```
  searchNonStandard = FALSE,
  includeDescendants = TRUE,
  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.
exclude	Character vector of words to identify concepts to exclude.
domains	Character vector with one or more of the OMOP CDM domain.
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.
includeAncestor	Either TRUE or FALSE. If TRUE the direct ancestor concepts of identified concepts will be included in the candidate codelist.

Value

tibble

Examples

```
## Not run:
cdm <- CodelistGenerator::mockVocabRef()
CodelistGenerator::getCandidateCodes(
  cdm = cdm,
  keywords = "osteoarthritis"
 )
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

getConceptClassId *getConceptClassId*

Description

getConceptClassId

Usage

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

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.
domain	Vocabulary domain

Value

The concept class used for a given set of domains

Examples

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

## End(Not run)
```

getDescendants *getDescendants*

Description

getDescendants

Usage

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

Arguments

cdm	cdm_reference via CDMConnector
conceptId	concpet_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 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

Examples

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

## End(Not run)
```

---

getDoseForm *getDoseForm*

Description

getDoseForm

Usage

```
getDoseForm(cdm)
```

Arguments

cdm	cdm_reference via CDMConnector

Value

The dose forms available for drug concepts

Examples

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

## End(Not run)
```

`getDrugIngredientCodes` *Get descendant codes for drug ingredients*

Description

Get descendant codes for drug ingredients

Usage

```
getDrugIngredientCodes(
  cdm,
  name = NULL,
  doseForm = NULL,
  ingredientRange = c(1, Inf),
  withConceptDetails = FALSE
)
```

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.
`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.
`withConceptDetails`	If FALSE, each item in the list of results (one per ingredient) will contain a vector of concept IDs for each ingredient. If TRUE each item in the list of results will contain a tibble with additional information on the identified concepts.

Value

A named list, with each item containing a vector of descendant concepts of an ingredient (if withConceptDetails was set as FALSE) or a tibble with the descendant concepts along with additional details about them (if withConceptDetails was set as TRUE).

Examples

```
## 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

```
getICD10StandardCodes(
  cdm,
  level = c("ICD10 Chapter", "ICD10 SubChapter"),
  name = NULL,
  includeDescendants = TRUE,
  withConceptDetails = FALSE
)
```

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.
`withConceptDetails`	If FALSE a vector of concept IDs will be returned for each ICD group If TRUE a tibble will be returned with additional information on the identified concepts.

Value

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

Examples

```
## Not run:
cdm <- mockVocabRef()
getICD10StandardCodes(cdm = cdm, level = c(
  "ICD10 Chapter",
  "ICD10 SubChapter"
))
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

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

```
  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

```
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

Examples

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

## End(Not run)
```

`getVocabularies` *getVocabularies*

Description

getVocabularies

Usage

```
getVocabularies(cdm)
```

Arguments

`cdm`	cdm_reference via CDMConnector

Value

Names of available vocabularies

Examples

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

## End(Not run)
```

`getVocabVersion` *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")
```

Arguments

backend	'database' (duckdb) or 'data_frame'

Value

cdm reference with mock vocabulary

Examples

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

## End(Not run)
```

`restrictToCodesInUse` *Filter a codelist to keep only the codes used in the database*

Description

Filter a codelist to keep only the codes used in the database

Usage

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

Arguments

x	A codelist
cdm	cdm_reference via CDMConnector
table	cdm table

Value

A list of integers indicating the codes used in the database

Examples

```
## 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 *Get source codes used in the database*

Description

Get source codes used in the database

Usage

```
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

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

## End(Not run)
```

summariseAchillesCodeUse *Summarise code use from achilles counts*

Description

Summarise code use from achilles counts

Usage

```
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	\ifelse{html}{\href{https://lifecycle.r-lib.org/articles/stages.html#deprecated}{\fi

Value

A tibble with results

Examples

```
## 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

```
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	\ifelse{html}{\href{https://lifecycle.r-lib.org/articles/stages.html#deprecated}{\fi

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")
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,
  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	\ifelse{html}{\href{https://lifecycle.r-lib.org/articles/stages.html#deprecated}{\fi

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*

Description

Find orphan codes related to a codelist

Usage

```
summariseOrphanCodes(
  x,
  cdm,
  domains = "Condition",
  standardConcept = "Standard",
  searchInSynonyms = TRUE,
  searchNonStandard = TRUE,
  includeDescendants = TRUE,
  includeAncestor = TRUE,
  minCellCount = lifecycle::deprecated()
)
```

Arguments

`x`	Codes for which to find codes related but not included (orphan codes)
`cdm`	cdm_reference via CDMConnector
`domains`	Character vector with one or more of the OMOP CDM domain.
`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.

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

minCellCount
: \ifelse{html}{\href{https://lifecycle.r-lib.org/articles/stages.html#deprecated}{\fi

Value

A codelist containing code related to (but not in) the target codelist that are present used in the cdm

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,
domains = "Condition",
standardConcept = "Standard",
searchInSynonyms = FALSE,
searchNonStandard = FALSE,
includeDescendants = TRUE,
includeAncestor = FALSE)

orphan_codes
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

---

tableAchillesCodeUse *Format the result of summariseAchillesCodeUse into a visual table.*

Description

[Experimental]

Usage

```
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

```
## Not run:
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)

## End(Not run)
```

tableCodeUse *Format the result of summariseCodeUse into a visual table.*

Description

[Experimental]

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

```
## 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)
tableCodeUse(results)
CDMConnector::cdmDisconnect(cdm)

## End(Not run)
```

`tableCohortCodeUse` *Format the result of summariseCohortCodeUse into a visual table.*

Description

[Experimental]

Usage

```
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)
```

---

tableOrphanCodes *Format the result of summariseOrphanCodes into a visual table.*

Description

[Experimental]

Usage

```
tableOrphanCodes(
  result,
  type = "gt",
```

```
  header = c("cdm_name", "estimate"),
  conceptId = TRUE,
  standard = TRUE,
  vocabulary = TRUE,
  relationship = TRUE,
  groupColumns = NULL,
  settings = character(),
  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.
relationship	If TRUE relationship id will be displayed.
groupColumns	Columns to use as group labels. Allowed columns are cdm_name and/or codelist_name.
settings	Vector with the settings columns to display.
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

```
## 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,
```

```
  includeAncestor = FALSE)

  tableOrphanCodes(orphan_codes)

  CDMConnector::cdmDisconnect(cdm)

  ## End(Not run)
```

Index