

# Package: CodelistGenerator (via r-universe)

October 17, 2024

**Title** Identify Relevant Clinical Codes and Evaluate Their Use

**Version** 3.2.1

**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

**Roxygen** list(markdown = TRUE)

**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

**Repository** <https://darwin-eu.r-universe.dev>

**RemoteUrl** <https://github.com/darwin-eu/codelistgenerator>

**RemoteRef** HEAD

**RemoteSha** c849af4b5dfa7bf7a00ca28ce1f09e1c1d178b77

## Contents

availableATC . . . . .	3
availableICD10 . . . . .	3
availableIngredients . . . . .	4
codesFromCohort . . . . .	4
codesFromConceptSet . . . . .	5
codesInUse . . . . .	5
compareCodelists . . . . .	6
doseFormToRoute . . . . .	7
getATCCodes . . . . .	7
getCandidateCodes . . . . .	8
getConceptClassId . . . . .	10
getDescendants . . . . .	10
getDomains . . . . .	11
getDoseForm . . . . .	12
getDoseUnit . . . . .	13
getDrugIngredientCodes . . . . .	13
getICD10StandardCodes . . . . .	14
getMappings . . . . .	15
getRelationshipId . . . . .	16
getRouteCategories . . . . .	17
getVocabularies . . . . .	18
getVocabVersion . . . . .	18
mockVocabRef . . . . .	19
restrictToCodesInUse . . . . .	19
sourceCodesInUse . . . . .	20
stratifyByConcept . . . . .	21
stratifyByDoseUnit . . . . .	21
stratifyByRouteCategory . . . . .	22
subsetOnDoseUnit . . . . .	22
subsetOnRouteCategory . . . . .	23
subsetToCodesInUse . . . . .	23
summariseAchillesCodeUse . . . . .	24
summariseCodeUse . . . . .	25
summariseCohortCodeUse . . . . .	26
summariseOrphanCodes . . . . .	28
tableAchillesCodeUse . . . . .	29
tableCodeUse . . . . .	30
tableCohortCodeUse . . . . .	31
tableOrphanCodes . . . . .	33
<b>Index</b>	<b>35</b>

---

availableATC	<i>Get all ATC codes from the cdm</i>
--------------	---------------------------------------

---

**Description**

Get all ATC codes from the cdm

**Usage**

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

**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"

**Value**

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

**Examples**

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

---

availableICD10	<i>Get all ICD codes from the cdm</i>
----------------	---------------------------------------

---

**Description**

Get all ICD codes from the cdm

**Usage**

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

**Arguments**

cdm	cdm_reference via CDMConnector
level	Can be either "ICD10 Chapter" or "ICD10 SubChapter"

**Value**

A vector list of all ICD10 codes for the chosen level(s) found in the concept table of cdm.

**Examples**

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

---

availableIngredients *Get all ingredients codes from the cdm*

---

**Description**

Get all ingredients codes from the cdm

**Usage**

```
availableIngredients(cdm)
```

**Arguments**

cdm                      cdm\_reference via CDMConnector

**Value**

A vector list of all ingredient level codes found in the concept table of cdm.

**Examples**

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

---

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

**Examples**

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

---

codesInUse     *Use achilles counts to get codes used in the database*

---

**Description**

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**

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

---

compareCodelists	<i>Compare two codelists</i>
------------------	------------------------------

---

**Description**

Compare two codelists

**Usage**

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

**Arguments**

codelist1	Output of getCandidateCodes
codelist2	Output of getCandidateCodes

**Value**

tibble

**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
)

```

---

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.
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**

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

---

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**

```

cdm <- CodelistGenerator::mockVocabRef()
CodelistGenerator::getCandidateCodes(
  cdm = cdm,

```

```

    keywords = "osteoarthritis"
  )

```

---

getConceptClassId	<i>getConceptClassId</i>
-------------------	--------------------------

---

### 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

```

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

```

---

getDescendants	<i>getDescendants</i>
----------------	-----------------------

---

### Description

getDescendants

**Usage**

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

**Arguments**

cdm	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 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**

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

---

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**

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

---

`getDoseForm`

*getDoseForm*

---

**Description**

`getDoseForm`

**Usage**

```
getDoseForm(cdm)
```

**Arguments**

`cdm`                    `cdm_reference` via CDMConnector

**Value**

The dose forms available for drug concepts

**Examples**

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

---

getDoseUnit	<i>Get available routes in a cdm reference.</i>
-------------	---

---

**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	<i>Get descendant codes for drug ingredients</i>
------------------------	--

---

**Description**

Get descendant codes for 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	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.
nameStyle	Name style to apply to returned list. Can be one of "{concept_code}_{concept_name}", "{concept_code}", or "{concept_name}".
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"

**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 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,
  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

**Examples**

```
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 cdm\_reference via CDMConnector::cdm\_from\_con()

nonStandardVocabularies  
Character vector

**Value**

tibble

**Examples**

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

---

`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**

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

---

getRouteCategories      *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**

```
getRouteCategories(cdm)
```

**Arguments**

cdm                    A cdm reference.

**Value**

A character vector with available routes

---

`getVocabularies`      *getVocabularies*

---

**Description**

`getVocabularies`

**Usage**

```
getVocabularies(cdm)
```

**Arguments**

`cdm`                      `cdm_reference` via CDMConnector

**Value**

Names of available vocabularies

**Examples**

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

---

`getVocabVersion`      *getVocabVersion*

---

**Description**

`getVocabVersion`

**Usage**

```
getVocabVersion(cdm)
```

**Arguments**

`cdm`                      `cdm_reference` via CDMConnector

**Value**

the vocabulary version being used

**Examples**

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

---

mockVocabRef	<i>Generate example vocabulary database</i>
--------------	---

---

**Description**

Generate example vocabulary database

**Usage**

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

**Arguments**

backend            'database' (duckdb) or 'data\_frame'

**Value**

cdm reference with mock vocabulary

**Examples**

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

---

restrictToCodesInUse	<i>Use achilles counts to filter a codelist to keep only the codes used in the database</i>
----------------------	---

---

**Description**

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

**Usage**

```
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

**Examples**

```
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)
```

---

sourceCodesInUse	<i>Use achilles counts to get source codes used in the database</i>
------------------	---

---

**Description**

Use achilles counts to 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**

```

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

```

---

stratifyByConcept	<i>Stratify a codelist by the concepts included within it</i>
-------------------	---

---

**Description**

Stratify a codelist by the concepts included within it

**Usage**

```
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	<i>Stratify a codelist by dose unit</i>
--------------------	---

---

**Description**

Stratify a codelist by dose unit

**Usage**

```
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 <code>getDoseUnit()</code> 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**

```
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**

```
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

**Value**

Use achilles counts to filter codelist to 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"),
  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	<code>\ifelse{html}{\href{https://lifecycle.r-lib.org/articles/stages.html#deprecated}}{\fi}</code>

**Value**

A tibble with results



**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,
  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	<code>\ifelse{html}{\href{https://lifecycle.r-lib.org/articles/stages.html#deprecated}}{\fi}</code>

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

## 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

```
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)
```

---

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"),
  groupColumns = character(),
  hide = character(),
  .options = list(),
  conceptId = lifecycle::deprecated(),
  standard = lifecycle::deprecated(),
  vocabulary = lifecycle::deprecated(),
  excludeColumns = lifecycle::deprecated()
)
```

### 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.
groupColumns	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.
conceptId	Deprecated.
standard	Deprecated.
vocabulary	Deprecated.
excludeColumns	Deprecated.

**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	<i>Format the result of summariseCodeUse into a table.</i>
--------------	--

---

**Description**

Format the result of summariseCodeUse into a table.

**Usage**

```
tableCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumns = character(),
  hide = character(),
  .options = list(),
  splitStrata = lifecycle::deprecated(),
  conceptId = lifecycle::deprecated(),
  sourceConcept = lifecycle::deprecated(),
  excludeColumns = lifecycle::deprecated()
)
```

**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.

groupColumns	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.
splitStrata	Deprecated.
conceptId	Deprecated.
sourceConcept	Deprecated.
excludeColumns	Deprecated.

**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	<i>Format the result of summariseCohortCodeUse into a table.</i>
--------------------	--

---

**Description**

Format the result of summariseCohortCodeUse into a table.

**Usage**

```
tableCohortCodeUse(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumns = NULL,
  timing = FALSE,
  hide = character(),
  .options = list(),
  excludeColumns = lifecycle::deprecated(),
  splitStrata = lifecycle::deprecated(),
  conceptId = lifecycle::deprecated(),
  sourceConcept = 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 <code>visOmopResults::tableType()</code>
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.
groupColumns	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.
timing	If TRUE the timing setting will be displayed.
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. <code>visOmopResults::tableOptions()</code> shows allowed arguments and their default values.
excludeColumns	Deprecated.
splitStrata	Deprecated.
conceptId	Deprecated.
sourceConcept	Deprecated.

**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 table.*


---

**Description**

Format the result of summariseOrphanCodes into a table.

**Usage**

```
tableOrphanCodes(
  result,
  type = "gt",
  header = c("cdm_name", "estimate_name"),
  groupColumns = character(),
  hide = character(),
  .options = list(),
  conceptId = lifecycle::deprecated(),
  standard = lifecycle::deprecated(),
  vocabulary = lifecycle::deprecated(),
  excludeColumns = lifecycle::deprecated()
)
```

**Arguments**

result	A summarised result with results of the type "orphan_codes".
type	Type of desired formatted table. To see supported formats use <code>visOmopResults::tableType()</code>
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.
groupColumns	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. <code>visOmopResults::tableOptions()</code> shows allowed arguments and their default values.
conceptId	Deprecated.
standard	Deprecated.
vocabulary	Deprecated.
excludeColumns	Deprecated.

**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)
```

# Index

## \* datasets

- doseFormToRoute, [7](#)
  
- availableATC, [3](#)
- availableICD10, [3](#)
- availableIngredients, [4](#)
  
- codesFromCohort, [4](#)
- codesFromConceptSet, [5](#)
- codesInUse, [5](#)
- compareCodelists, [6](#)
  
- doseFormToRoute, [7](#)
  
- getATCCodes, [7](#)
- getCandidateCodes, [8](#)
- getConceptClassId, [10](#)
- getDescendants, [10](#)
- getDomains, [11](#)
- getDoseForm, [12](#)
- getDoseUnit, [13](#)
- getDrugIngredientCodes, [13](#)
- getICD10StandardCodes, [14](#)
- getMappings, [15](#)
- getRelationshipId, [16](#)
- getRouteCategories, [17](#)
- getVocabularies, [18](#)
- getVocabVersion, [18](#)
  
- mockVocabRef, [19](#)
  
- restrictToCodesInUse, [19](#)
  
- sourceCodesInUse, [20](#)
- stratifyByConcept, [21](#)
- stratifyByDoseUnit, [21](#)
- stratifyByRouteCategory, [22](#)
- subsetOnDoseUnit, [22](#)
- subsetOnRouteCategory, [23](#)
- subsetToCodesInUse, [23](#)
- summariseAchillesCodeUse, [24](#)

- summariseCodeUse, [25](#)
- summariseCohortCodeUse, [26](#)
- summariseOrphanCodes, [28](#)
  
- tableAchillesCodeUse, [29](#)
- tableCodeUse, [30](#)
- tableCohortCodeUse, [31](#)
- tableOrphanCodes, [33](#)