

Package ‘CodelistGenerator’

May 31, 2023

Title Generate Code Lists for the OMOP Common Data Model

Version 1.3.0

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 7.2.3

Imports CDMConnector (>= 0.5.1), checkmate (>= 2.0.0), DBI (>= 1.1.0), dplyr (>= 1.0.0), magrittr (>= 2.0.0), rlang (>= 1.0.0), glue (>= 1.5.0), stringr (>= 1.4.0), tidyselect (>= 1.2.0), tidyr (>= 1.2.0), cli (>= 3.1.0)

Suggests arrow (>= 9.0.0), covr, dbplyr (>= 2.2.1), knitr, readr (>= 2.1.0), duckdb, DT, rmarkdown, here (>= 1.0.0), testthat (>= 3.0.0), kableExtra (>= 1.0.0), RPostgres, odbc, spelling

Config/testthat/edition 3

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>>)

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

Repository CRAN

Date/Publication 2023-05-30 22:40:02 UTC

R topics documented:

compareCodelists	2
getATCCodes	3
getCandidateCodes	4

getConceptClassId	5
getDescendants	6
getDomains	7
getDoseForm	7
getDrugIngredientCodes	8
getICD10StandardCodes	9
getMappings	9
getVocabularies	10
getVocabVersion	11
mockVocabRef	11

Index	12
--------------	-----------

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  
  )  
  DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

`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)
```

Arguments

<code>cdm</code>	cdm_reference via CDMConnector
<code>level</code>	ATC level. Can be one or more of "ATC 1st", "ATC 2nd", "ATC 3rd", "ATC 4th", and "ATC 5th"
<code>name</code>	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.
<code>doseForm</code>	Only descendants codes with the specified dose form will be returned. If NULL, descendant codes will be returned regardless of dose form.

Value

A named list, with each element containing the descendant concepts for a particular ATC group

Examples

```
cdm <- mockVocabRef()  
getATCCodes(cdm = cdm, level = "ATC 1st")  
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

<code>getCandidateCodes</code>	<i>Generate candidate codelist for the OMOP CDM</i>
--------------------------------	---

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",
  conceptClassId = NULL,
  doseForm = NULL,
  vocabularyId = NULL,
  standardConcept = "Standard",
  exactMatch = FALSE,
  searchInSynonyms = FALSE,
  searchViaSynonyms = FALSE,
  searchNonStandard = FALSE,
  includeSequela = FALSE,
  includeDescendants = TRUE,
  includeAncestor = FALSE,
  fuzzyMatch = FALSE,
  maxDistanceCost = 0.1,
  verbose = FALSE
)
```

Arguments

<code>cdm</code>	<code>cdm_reference</code> via <code>CDMConnector</code>
<code>keywords</code>	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.
<code>exclude</code>	Character vector of words to identify concepts to exclude.
<code>domains</code>	Character vector with one or more of the OMOP CDM domain.
<code>conceptClassId</code>	Character vector with one or more concept class of the Concept
<code>doseForm</code>	The dose form associated with a drug
<code>vocabularyId</code>	Character vector with one or more vocabulary of the Concept
<code>standardConcept</code>	Character vector with one or more of "Standard", "Classification", and "Non-standard". These correspond to the flags used for the <code>standard_concept</code> field in the concept table of the <code>cdm</code> .

exactMatch	Either TRUE or FALSE. If TRUE only exact matches of keywords will be identified when running the initial search.
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.
searchViaSynonyms	Either TRUE or FALSE. If TRUE the code will also search via the concept synonym table.
searchNonStandard	Either TRUE or FALSE. If TRUE the code will also search via non-standard concepts.
includeSequela	Either TRUE or FALSE. If TRUE, codes associated via a concept relationship of 'Due to of' or 'Occurs before' will also be identified.
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.
fuzzyMatch	Either TRUE or FALSE. If TRUE the fuzzy matching will be used, with approximate matches identified.
maxDistanceCost,	The maximum number/fraction of match cost (generalized Levenshtein distance) for fuzzy matching (see <code>base::agrep</code> for further details).
verbose	Either TRUE or FALSE. If TRUE, progress will be reported.

Value

tibble

Examples

```

cdm <- CodelistGenerator::mockVocabRef()
CodelistGenerator::getCandidateCodes(
  cdm = cdm,
  keywords = "osteoarthritis"
)
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)

```

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")
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

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

Description

getDescendants

Usage

```
getDescendants(cdm, conceptId, withAncestor = FALSE, 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 ";"
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)
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

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)
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

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)
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

`getDrugIngredientCodes`

Get descendant codes for drug ingredients

Description

Get descendant codes for drug ingredients

Usage

```
getDrugIngredientCodes(cdm, name = NULL, doseForm = NULL)
```

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.

Value

A named list, with each item containing descendant concepts of an ingredient

Examples

```
cdm <- mockVocabRef()
getDrugIngredientCodes(cdm = cdm, name = "Adalimumab")
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

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"))
```

Arguments

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

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"
))
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

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"
)
```

`getVocabularies` *getVocabularies*

Description

`getVocabularies`

Usage

```
getVocabularies(cdm)
```

Arguments

cdm cdm_reference via CDMConnector

Value

Names of available vocabularies

Examples

```
cdm <- mockVocabRef()
getVocabularies(cdm = cdm)
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

getVocabVersion	<i>getVocabVersion</i>
-----------------	------------------------

Description

getVocabVersion

Usage

```
getVocabVersion(cdm)
```

Arguments

cdm cdm_reference via CDMConnector

Value

the vocabulary version being used

Examples

```
cdm <- mockVocabRef()
getVocabVersion(cdm = cdm)
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

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

Description

Generate example vocabulary database

Usage

```
mockVocabRef(backend = "database")
```

Arguments

backend 'database' (duckdb), 'arrow' (parquet files), or 'data_frame'

Value

cdm reference with mock vocabulary

Examples

```
cdm <- mockVocabRef()
cdm
DBI::dbDisconnect(attr(cdm, "dbcon"), shutdown = TRUE)
```

Index

[compareCodelists](#), [2](#)

[getATCCodes](#), [3](#)

[getCandidateCodes](#), [4](#)

[getConceptClassId](#), [5](#)

[getDescendants](#), [6](#)

[getDomains](#), [7](#)

[getDoseForm](#), [7](#)

[getDrugIngredientCodes](#), [8](#)

[getICD10StandardCodes](#), [9](#)

[getMappings](#), [9](#)

[getVocabularies](#), [10](#)

[getVocabVersion](#), [11](#)

[mockVocabRef](#), [11](#)