Package 'CodelistGenerator'

May 7, 2026

Title Identify Relevant Clinical Codes and Evaluate Their Use

Version 4.0.2

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 7.3.3

Depends R (>= 4.1)

Imports DBI (>= 1.1.0), dplyr (>= 1.1.0), omopgenerics (>= 1.3.4), rlang (>= 1.0.0), glue (>= 1.5.0), stringr (>= 1.6.0), stringi (>= 1.8.1), tidyr (>= 1.2.0), cli (>= 3.1.0), purrr, clock, PatientProfiles (>= 1.4.4), vctrs, jsonlite, lifecycle

Suggests covr, duckdb, CDMConnector (>= 2.1.0), visOmopResults (>= 1.4.0), CohortConstructor (>= 0.5.0), knitr, rmarkdown, testthat (>= 3.0.0), RPostgres, odbc, spelling, tibble, gt, flextable, omock, tictoc

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

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

Language en-US

LazyData true

NeedsCompilation no

Author Edward Burn [aut, cre] (ORCID: <https://orcid.org/0000-0002-9286-1128>), Marta Alcalde-Herraiz [aut] (ORCID: <https://orcid.org/0009-0002-4405-1814>), Martí Català [aut] (ORCID: <https://orcid.org/0000-0003-3308-9905>), Xihang Chen [aut] (ORCID: <https://orcid.org/0009-0001-8112-8959>), Nuria Mercade-Besora [aut] (ORCID: <https://orcid.org/0009-0006-7948-3747>),

Mike Du [aut] (ORCID: <https://orcid.org/0000-0002-9517-8834>),
Danielle Newby [aut] (ORCID: <https://orcid.org/0000-0002-3001-1478>)

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

Repository CRAN

Date/Publication 2026-01-19 10:50:02 UTC

Contents

`addConcepts` *Add concepts to a codelist*

Description

Add concepts to a codelist

Usage

```
addConcepts(x, cdm, concepts, codelistName = NULL)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`concepts`	Concepts_ID to add
`codelistName`	Name or names of codelist in x. If NULL, all codelist present in x will be considered.

Value

A codelist

Examples

```
library(omock)
library(CDMConnector)

# Creating CDM object
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Creating codelist
codelist <- getDrugIngredientCodes(cdm,
                                   nameStyle = "{concept_name}")

# Add a concept to all the codelists:
codelist$acetaminophen
codelist <- codelist |>
 addConcepts(cdm, concepts = c(1L))
codelist$acetaminophen

# Add a concept to a specific codelist
codelist$amiodarone
codelist <- codelist |>
  addConcepts(cdm, concepts = c(2L), codelistName = "amiodarone")
codelist$amiodarone

# See function: `excludeConcepts()` for details on how to remove specific concepts
# from a codelist
```

`asCodelist` *Coerce to a codelist*

Description

Coerce to a codelist

Usage

```
asCodelist(x, ...)

## S3 method for class 'codelist'
asCodelist(x, ...)

## S3 method for class 'codelist_with_details'
asCodelist(x, ...)

## S3 method for class 'concept_set_expression'
asCodelist(x, cdm, ...)

## S3 method for class 'candidate_codes'
asCodelist(x, ...)
```

Arguments

x	Only codelist_with_details and candidate_codes are currently supported.
...	For extensibility
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

Value

codelist

Examples

```
library(omock)
library(CDMConnector)

# Creating CDM object
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Create codelist from a codelist_with_details
codelist <- getDrugIngredientCodes(cdm,
                                   name = "acetaminophen",
                                   nameStyle = "{concept_name}",
                                   type = "codelist_with_details")

asCodelist(codelist)

# Create codelist from a candidate_codes
codelist <- getCandidateCodes(cdm,
                              keywords = "arthritis")

asCodelist(codelist)
```

---

asCodelistWithDetails *Coerce to a codelist with details*

Description

Coerce to a codelist with details

Usage

```
asCodelistWithDetails(x, cdm, ...)

## S3 method for class 'codelist_with_details'
asCodelistWithDetails(x, ...)
```

```
## S3 method for class 'codelist'
asCodelistWithDetails(x, cdm, ...)

## S3 method for class 'candidate_codes'
asCodelistWithDetails(x, cdm, ...)
```

Arguments

x	Only codelist and candidate_codes are currently supported.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
...	For extensibility

Value

codelist

Examples

```
library(omock)
library(CDMConnector)

# Creating CDM object
path <- downloadMockDataset(datasetName = "GiBleed",
                            path = NULL,
                            overwrite = NULL)
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Create codelist_with_details from a codelist
codelist <- getDrugIngredientCodes(cdm,
                                   name = "acetaminophen",
                                   nameStyle = "{concept_name}",
                                   type = "codelist")

asCodelistWithDetails(codelist, cdm)

# Create codelist from a candidate_codes
codelist <- getCandidateCodes(cdm,
                              keywords = "arthritis")

asCodelistWithDetails(codelist)
```

asConceptSetExpression *Coerce to a concept set expression*

Description

Coerce to a concept set expression

Usage

```
asConceptSetExpression(x, ...)

## S3 method for class 'codelist'
asConceptSetExpression(x, ...)

## S3 method for class 'codelist_with_details'
asConceptSetExpression(x, ...)
```

Arguments

x	Codelist or codelist with details
...	For extensibility

Value

codelist

Examples

```
library(omock)
library(CDMConnector)

# Creating CDM object
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Create concept_set_expression from a codelist
codelist <- getDrugIngredientCodes(cdm,
                                   name = "acetaminophen",
                                   nameStyle = "{concept_name}",
                                   type = "codelist")

asConceptSetExpression(codelist)

# Create concept_set_expression from a codelist_with_details
codelist <- getDrugIngredientCodes(cdm,
                                   name = "acetaminophen",
                                  nameStyle = "{concept_name}",
                                   type = "codelist_with_details")
```

```
asConceptSetExpression(codelist)
```

`associatedConceptClassIds` *Get the concept classes associated with a codelist*

Description

Get the concept classes associated with a codelist

Usage

```
associatedConceptClassIds(x, cdm, standardConcept = "Standard", domain = NULL)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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 availableDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.

Value

The concept classes

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Get concept_class_ids in a codelist
x <- newCodelist(list("codes1" = c(1118088L, 40213201L, 35208414L),
                      "codes2" = c(1557272L, 4336464L, 4295880L)))
associatedConceptClassIds(x, cdm,
                          standardConcept = "Standard")
```

```
# Notice that this corresponds to the information provided by `concept_class_id`
# column in the `concept` table
```

`associatedDomains` *Get the domains associated with a codelist*

Description

Get the domains associated with a codelist

Usage

```
associatedDomains(x, cdm, standardConcept = "Standard")
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all domains available in a codelist
codelist <- newCodelist(list("codes1" = c(194152L, 1830279L, 40558872L),
                             "codes2" = c(44022939L)))
associatedDomains(x = codelist, cdm = cdm,
                  standardConcept = c("Non-standard", "Standard"))
```

associatedDoseForms *Get the dose forms associated with drug concepts in a codelist*

Description

Get the dose forms associated with drug concepts in a codelist

Usage

```
associatedDoseForms(x, cdm)
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

Value

The dose forms available for drug concepts.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all dose forms available in a codelist
codelist <- newCodelist(list("codes1" = c(194152L, 1830279L, 40558872L),
                             "codes2" = c(44022939L)))
associatedDoseForms(x = codelist, cdm = cdm)
```

associatedDoseUnits *Get available dose units*

Description

Get the dose units associated with a codelist

Usage

```
associatedDoseUnits(x, cdm, standardConcept = "Standard")
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 character vector with available routes.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

codelist <- newCodelist(list("codes1" = c(194152L, 1830279L, 40558872L),
                             "codes2" = c(44022939L, 1830282L)))
associatedDoseUnits(cdm = cdm,
                    x = codelist)
```

---

associatedDrugIngredients

Get the names of drug ingredients associated with codelist

Description

Get the names of drug ingredients associated with codelist

Usage

```
associatedDrugIngredients(x, cdm, standardConcept = "Standard")
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

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

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all drug ingredients associated with a codelist
codelist <- newCodelist(list("codes1" = c(37498042L),
                             "codes2" = c( 42899580L, 35741956L)))
associatedDrugIngredients(x = codelist, cdm = cdm,
                          standardConcept = c("Standard", "Non-standard"))
```

---

associatedRelationshipIds

Get available relationships with concepts in a codelist

Description

Get available relationships with concepts in a codelist

Usage

```
associatedRelationshipIds(
  x,
  cdm,
  standardConcept1 = "Standard",
  standardConcept2 = "Standard",
  domains1 = "Condition",
  domains2 = "Condition"
)
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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. If NULL, all domains are considered.
domains2	Character vector with one or more of the OMOP CDM domain. If NULL, all domains are considered.

Value

A character vector with unique concept relationship values.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()


codelist <- newCodelist(list("codes1" = c(8479L, 4117795L),
                             "codes2" = c(8480L, 8600L, 8481L, 4189167L)))
associatedRelationshipIds(x = codelist, cdm = cdm,
                      standardConcept1 = c("Standard", "Non-standard", "Classification"),
                      standardConcept2 = c("Standard", "Non-standard", "Classification"),
                          domains1 = NULL,
                          domains2 = NULL)
```

---

associatedRouteCategories

Get drug routes associated with a codelist

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

```
associatedRouteCategories(x, cdm)
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

Value

A character vector with all available routes.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all dose forms available in a codelist
codelist <- newCodelist(list("codes1" = c(194152L, 1830279L, 40558872L),
                             "codes2" = c(44022939L)))
```

---

associatedVocabularies

Get the vocabularies associated with a codelist

Description

Get the vocabularies associated with a codelist

Usage

```
associatedVocabularies(x, cdm, standardConcept = "Standard", domain = NULL)
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

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 availableDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.

Value

Names of available vocabularies.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all vocabularies from a codelist
codelist <- newCodelist(list("codes1" = c(35604877L, 35604394L),
                             "codes2" = c(4214687L)))
associatedVocabularies(cdm = cdm,
                       x = codelist)
```

---

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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

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

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get ATC 1st level classification codes
availableATC(cdm, level = "ATC 1st")

# Get all ATC classification codes
availableATC(cdm, level = c("ATC 1st", "ATC 2nd", "ATC 3rd", "ATC 4th", "ATC 5th"))
```

---

availableConceptClassIds

Get the available concept classes used in a given set of domains

---

Description

Get the available concept classes used in a given set of domains

Usage

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

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 availableDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.

Value

The concept classes

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Get all available concept_class_ids in the CDM
availableConceptClassIds(cdm,
                         standardConcept = "Standard")

# Get all available concept_class_ids in the CDM for a specific domain
availableConceptClassIds(cdm,
                         standardConcept = "Standard",
                         domain = "Condition")

# Notice that this corresponds to the information provided by `concept_class_id`
# column in the `concept` table
```

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

Description

Get the domains available in the cdm

Usage

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

Arguments

`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all domains available in the CDM for standard concepts
availableDomains(cdm = cdm, standardConcept = "Standard")
```

`availableDoseForms` *Get the dose forms for drug concepts*

Description

Get the dose forms for drug concepts

Usage

```
availableDoseForms(cdm)
```

Arguments

`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

Value

The dose forms available for drug concepts.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all domains available in the CDM
availableDoseForms(cdm = cdm)
```

availableDoseUnits *Get available dose units*

Description

Get the available dose units

Usage

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

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 character vector with available routes.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all dose units available in the CDM
availableDoseUnits(cdm = cdm)
```

availableDrugIngredients *Get the names of all available drug ingredients*

Description

Get the names of all available drug ingredients

Usage

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

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 containing the concept names for all ingredient level codes found in the concept table of cdm.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all drug ingredients available in the CDM for standard concepts
availableDrugIngredients(cdm = cdm)
```

---

availableRelationshipIds

Get available relationships between concepts

Description

Get available relationships between concepts

Usage

```
availableRelationshipIds(
  cdm,
  standardConcept1 = "Standard",
  standardConcept2 = "Standard",
  domains1 = "Condition",
  domains2 = "Condition"
)
```

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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. If NULL, all domains are considered.
domains2	Character vector with one or more of the OMOP CDM domain. If NULL, all domains are considered.

Value

A character vector with unique concept relationship values.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all relationship ids in the CDM between `Condition` and `Standard` concepts.
availableRelationshipIds(cdm = cdm,
                         standardConcept1 = "Standard",
                         standardConcept2 = "Standard",
                         domains1 = "Condition",
                         domains2 = "Condition")
```

---

availableRouteCategories

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

```
availableRouteCategories(cdm)
```

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

Value

A character vector with all available routes.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all domains available in the CDM
availableRouteCategories(cdm = cdm)
```

---

availableVocabularies *Get the available vocabularies available in the cdm*

Description

Get the available vocabularies available in the cdm

Usage

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

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 availableDomains(). If NULL, all supported domains are included: Condition, Drug, Procedure, Device, Observation, and Measurement.

Value

Names of available vocabularies.

Examples

```
library(CodelistGenerator)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Get all vocabularies available in the CDM
availableVocabularies(cdm)

# Get all vocabularies available in the CDM for `Standard` and `Condition` concepts
availableVocabularies(cdm,
                      standardConcept = "Standard",
                      domain = "Condition")
```

`benchmarkCodelistGenerator` *Run benchmark of codelistGenerator analyses*

Description

Run benchmark of codelistGenerator analyses

Usage

```
benchmarkCodelistGenerator(cdm)
```

Arguments

`cdm`	a CDM reference object

Value

a tibble with time taken for different analysis

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()

timings <- benchmarkCodelistGenerator(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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
type	Can be "codelist", "codelist_with_details" or "concept_set_expression".

Value

Named list with concept_ids for each concept set.

Examples

```
library(CodelistGenerator)
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* **[Deprecated]**

Description

Get concept ids from JSON files containing concept sets **[Deprecated]**

Usage

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

Arguments

path	Path to a file or folder containing JSONs of concept sets.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
type	Can be "codelist", "codelist_with_details" or "concept_set_expression".

Value

Named list with concept_ids for each concept set.

Examples

```
library(CodelistGenerator)
library(omock)

# Create a CDM object
cdm <- mockCdmReference()

# Load JSON files
x <- codesFromConceptSet(cdm = cdm,
                         path =  system.file(package = "CodelistGenerator",
                                             "concepts_for_mock"))
x

# Load JSON files as codelist_with_details
x <- codesFromConceptSet(cdm = cdm,
                         path =  system.file(package = "CodelistGenerator",
                                             "concepts_for_mock"),
                         type = "codelist_with_details")
x

# Load JSON files as concept_set_expression
x <- codesFromConceptSet(cdm = cdm,
                         path =  system.file(package = "CodelistGenerator",
                                             "concepts_for_mock"),
                         type = "concept_set_expression")
x
```

---

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

```
library(CodelistGenerator)
library(omock)

# Create a CDM object
downloadMockDataset(datasetName = "GiBleed",
                    path = NULL,
                    overwrite = NULL)
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Compare two candidate_codes object
codes1 <- getCandidateCodes(
  cdm = cdm,
  keywords = "Arthritis",
  domains = "Condition",
  includeDescendants = TRUE)

codes2 <- getCandidateCodes(
  cdm = cdm,
  keywords = c("osteo"),
  domains = "Condition",
  includeDescendants = TRUE)

compareCodelists(
  codelist1 = codes1,
  codelist2 = codes2)

# Compare two codelists
acetaminophen <- getDrugIngredientCodes(cdm,
                                        name = "acetaminophen",
                                        nameStyle = "{concept_name}",
                                        type = "codelist")

hydrocodone <- getDrugIngredientCodes(cdm,
                                      name = "hydrocodone",
                                      nameStyle = "{concept_name}",
                                      type = "codelist")
compareCodelists(
```

```
  codelist1 = acetaminophen,
  codelist2 = hydrocodone)
# Notice that concept_name = NA as `codelist` class does not store this information
# for each concept.

# Compare two codelists_with_details
acetaminophen <- getDrugIngredientCodes(cdm,
                                        name = "acetaminophen",
                                        nameStyle = "{concept_name}",
                                        type = "codelist_with_details")

hydrocodone <- getDrugIngredientCodes(cdm,
                                      name = "hydrocodone",
                                      nameStyle = "{concept_name}",
                                      type = "codelist_with_details")
compareCodelists(
  codelist1 = acetaminophen,
  codelist2 = hydrocodone)
```

`doseFormToRoute` *Table showing the route category associated with each dose form.*

Description

Table showing the route category associated with each dose form.

Usage

```
doseFormToRoute
```

Format

A data frame

dose_form_concept_id Concept ID of each dose form

dose_form_concept_name Concept name of each dose form

route_category Route category associated to the dose form

Examples

```
library(CodelistGenerator)
doseFormToRoute
```

`excludeConcepts` *Exclude concepts from a codelist*

Description

Exclude concepts from a codelist

Usage

```
excludeConcepts(x, cdm, concepts, codelistName = NULL)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`concepts`	Concepts_id to exclude
`codelistName`	Name or names of codelist in x. If NULL, all codelist present in x will be considered.

Value

A codelist

Examples

```
library(omock)
library(CDMConnector)

# Creating CDM object
# downloadMockDataset(datasetName = "GiBleed")
cdm <- mockCdmFromDataset(datasetName = "GiBleed")

# Creating codelist
codelist <- getDrugIngredientCodes(cdm,
                                   nameStyle = "{concept_name}")

# Exclude concept to all the codelists:
codelist$acetaminophen
codelist <- codelist |>
  excludeConcepts(cdm, concepts = c(1125315L))
codelist$acetaminophen
# Add a concept to a specific codelist
codelist$amiodarone
codelist <- codelist |>
  excludeConcepts(cdm, concepts = c(1310034L), codelistName = "amiodarone")
codelist$amiodarone
```

```
# See function: `addConcepts()` for details on how to add specific concepts
# to a codelist
```

`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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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 'doseForms()' 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 'availableDoseUnits()' 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)
library(omock)

# Create CDM object
cdm <- mockCdmReference()

# Create a codelist with 1st level ATC codes available in the CDM
codelist <- getATCCodes(cdm = cdm,
                        level = "ATC 1st")
codelist

# Tune the name of the generated codelists
codelist <- getATCCodes(cdm = cdm,
                        level = "ATC 1st",
                        nameStyle = "{concept_name}_{concept_code}")
codelist

# Search for a specific ATC name of interest
codelist <- getATCCodes(cdm = cdm,
                        level = "ATC 2nd",
                        name = "immunostimulants")
codelist

# Restrict concepts to specific dose forms, dose units, or route categories.
# Remember that you can use `availableDoseForm()`, `availableDoseUnit()` and
# `availableRouteCategory()` to explore your codelist.
codelist <- getATCCodes(cdm = cdm,
                        level = "ATC 2nd",
                        doseForm = NULL,
                        doseUnit = NULL,
                        routeCategory = NULL,)
codelist

# You can also create directly a codelist_with_details using the argument `type`
codelist <- getATCCodes(cdm = cdm,
                        level = "ATC 1st",
                        type = "codelist_with_details")
codelist
```

`getCandidateCodes` *Perform a systematic search to identify a candidate codelist using the OMOP CDM vocabulary tables.*

Description

Based on the given search strategy, this function identifies a set of codes that may represent a clinical event of interest in data mapped to the OMOP CDM. These codes can then be considered for creating a study phenotype.

Usage

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

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
keywords	Character vector of terms to search for. • The search performed is broad, matching the provided keywords as substrings anywhere within concept names. For example, `keywords = c("sep")` will find "sepsis", "aseptic necrosis", and "acquired cardiac septal defect" (along with many more). • Where more than one word is given, all combinations of those words will be identified. For example, `keywords = c("knee osteoarthritis")` will identify a concept with the name "osteoarthritis of knee". • Multiple keywords can be provided. For example, `keywords = c("knee osteoarthritis", "hip osteoarthritis")` would identify both "osteoarthritis of knee" and "osteoarthritis of hip"
exclude	Character vector of words to identify concepts to exclude. For example, `getCandidateCodes(cdm, keywords = "septic", exclude = "aseptic", domains = "condition")` would remove concepts "aseptic" when seaching for concepts with "septic" in their name. • When one term contains multiple words (e.g., "knee osteoarthritis"), each word will be search individually, so that "osteoarthritis of knee" would also be excluded. If you only want to exclude partial matching terms, please add "/" at the beginning and the end of each term (e.g., `"/knee osteoarthritis/"`). Notice that, with this options, concepts like "rightknee osteoarthritis" will also be excluded (as this is a partial match), but "osteoarthritis of knee" won't be excluded. Different terms can have different rules (e.g., c("hip osteoarthritis", "/knee osteoarthritis/")).

- With multiple words, if we want exact matches accounting for word boundaries, we need to use /\b at the beginning and at the end of each expression. In the previous example, using "/bknee osteoarthritis/\b", "rightknee osteoarthritis" won't be excluded, but "History of knee osteoarthritis" will be excluded.

domains
: Character vector with one or more of the OMOP CDM domain for which to search within. If NULL, all domains are included in the search. Use availableDomains(cdm = cdm) to identify available domains to search within.

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 "candidate_codes" object. This includes a tibble with the potential codes of interest, along with an attribute containing the search strategy.

Examples

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

---

getDescendants *Get descendant codes for a given concept*

Description

Get descendant codes for a given concept

Usage

```
getDescendants(cdm, conceptId, withAncestor = FALSE)
```

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
conceptId	concept_id to search
withAncestor	If TRUE, return column with ancestor. In case of multiple ancestors, concepts will be separated by ";".

Value

The descendants of a given concept id.

Examples

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

---

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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 'doseForms()' 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 'availableDoseUnits()' 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

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

---

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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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"
)
```

---

intersectCodelists *Generate a codelist from the intersection of different codelists. The generated codelist will come out in alphabetical order.*

Description

Generate a codelist from the intersection of different codelists. The generated codelist will come out in alphabetical order.

Usage

```
intersectCodelists(x, keepOriginal = FALSE)
```

Arguments

x	A codelist.
keepOriginal	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

A codelist

Examples

```
library(CodelistGenerator)
library(omock)

# Create a CDM object
cdm <- mockCdmReference()

# Intersect two codelists
codelist <- newCodelist(list("mood" = c(37110496L, 4226696L, 4304866L),
                             "manic" = c(37110496L, 4226696L)))

intersectCodelists(codelist, keepOriginal = TRUE)


# Intersect two codelists_with_details
codelist <- asCodelistWithDetails(codelist, cdm)

intersectCodelists(codelist, keepOriginal = FALSE)
```

---

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
```

`searchStrategy` *Report the search strategy used to identify codes when using the* `getCandidateCodes()` *function*

Description

Report the search strategy used to identify codes when using the `getCandidateCodes()` function

Usage

```
searchStrategy(x)
```

Arguments

x	A codelist.

Value

A tibble with the search strategy

Examples

```
library(omock)
library(CodelistGenerator)
library(dplyr, warn.conflicts = FALSE)

# Create CDM object
cdm <- mockCdmFromDataset(datasetName = "GiBleed")
codes <- getCandidateCodes(cdm = cdm,
                           keywords = c("sprain", "fracture"),
                           exclude = "knee",
                           domains = "Condition",
                           standardConcept = "Standard",
                           searchNonStandard = FALSE,
                           searchInSynonyms = TRUE,
                           includeDescendants = TRUE,
                           includeAncestor = FALSE)

searchStrategy(codes) |>
```

```
glimpse()
```

`stratifyByBrand` *Stratify a codelist by brand category.*

Description

Stratify a codelist by brand category.

Usage

```
stratifyByBrand(
  x,
  cdm,
  nameStyle = "{codelist_name}_{brand}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{brand}` to include the brand name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

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

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()
codes <- newCodelist(list(
  concepts_1 = c(20L, 21L, 22L),
  concepts_2 = c(10L, 13L, 21L)
))

new_codes <- stratifyByBrand(x = codes,
                             cdm = cdm,
                             keepOriginal = TRUE)
new_codes
```

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

Description

Stratify a codelist by the concepts included within it.

Usage

```
stratifyByConcept(
  x,
  cdm,
  nameStyle = "{codelist_name}_{concept}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{concept}` to include the concept name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (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 <- newCodelist(list("concepts" = c(20L, 21L)))

new_codes <- stratifyByConcept(x = codes,
                               cdm = cdm,
                               keepOriginal = TRUE)

new_codes
```

`stratifyByDomain` *Stratify a codelist by domain category.*

Description

Stratify a codelist by domain category.

Usage

```
stratifyByDomain(
  x,
  cdm,
  nameStyle = "{codelist_name}_{domain}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{domain}` to include the domain name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

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

Examples

```
library(CodelistGenerator)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- newCodelist(list("concepts_1" = c(20L,21L,22L),
                          "concepts_2" = c(10L,13L,21L)))
new_codes <- stratifyByDomain(x = codes,
                              cdm = cdm,
                              keepOriginal = TRUE)
new_codes
```

`stratifyByDoseForm` *Stratify a codelist by dose form.*

Description

Stratify a codelist by dose form.

Usage

```
stratifyByDoseForm(
  x,
  cdm,
  nameStyle = "{codelist_name}_{dose_form}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{dose_form}` to include the dose form name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

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

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()

codes <- newCodelist(list("codes" = c(10L, 20L, 21L)))
new_codes <- stratifyByDoseForm(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,
  nameStyle = "{codelist_name}_{dose_unit}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{dose_unit}` to include the dose unit name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

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

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()

codes <- newCodelist(list("concepts" = c(20L, 21L)))
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,
  nameStyle = "{codelist_name}_{route_category}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{route_category}` to include the route category name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

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

Examples

```
library(CodelistGenerator)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- newCodelist(list("concepts" = c(20,21,22)))
new_codes <- stratifyByRouteCategory(x = codes,
                                     cdm = cdm,
                                     keepOriginal = TRUE)
new_codes
```

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

Description

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

Usage

```
stratifyByVocabulary(
  x,
  cdm,
  nameStyle = "{codelist_name}_{vocabulary}",
  keepOriginal = FALSE
)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`nameStyle`	Naming of the new codelists, use `{codelist_name}` to include the codelist name and `{vocabulary}` to include the vocabulary name.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

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

Examples

```
library(CodelistGenerator)

cdm <- mockVocabRef()

codes <- stratifyByVocabulary(
  x = newCodelist(list("codes" = c(10L, 13L, 15L))),
  cdm = cdm,
  keepOriginal = TRUE
)

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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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 availableDomains(). 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)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- subsetOnDomain(
              x = newCodelist(list("codes" = c(10,13,15))),
              cdm = cdm,
              domain = "Drug")
codes
```

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

Description

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

Usage

```
subsetOnDoseForm(x, cdm, doseForm, negate = FALSE)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`doseForm`	Dose form/s. See `availableDoseForms()` to explore available dose forms in your codelist.
`negate`	If FALSE, only concepts with the dose form specified will be returned. If TRUE, concepts with the dose form specified will be excluded.

Value

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

Examples

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

codelist <- newCodelist(list("codes" = c(10L,20L,21L)))

# Dose forms present in our codelist:
codelist |> associatedDoseForms(cdm)

codes <- subsetOnDoseForm(
              x = codelist,
              cdm = cdm,
              doseForm = "Injection")
codes

codes |> associatedDoseForms(cdm)
```

`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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`doseUnit`	Only codes with the specified dose unit will be returned. If NULL, descendant codes will be returned regardless of dose unit Use 'availableDoseUnits()' 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)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- subsetOnDoseUnit(x = newCodelist(list("codes" = c(20,21))),
                          cdm = cdm,
                          doseUnit = c("milligram"))

codes
```

`subsetOnIngredientRange` *Subset a codelist to only those codes with a range of number of ingredients*

Description

Subset a codelist to only those codes with a range of number of ingredients

Usage

```
subsetOnIngredientRange(x, cdm, ingredientRange, negate = FALSE)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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.
`negate`	If FALSE, only concepts with the ingredient range specified will be returned (both limits included). If TRUE, concepts with number of ingredients outside the range will be returned.

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)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- subsetOnIngredientRange(
              x = newCodelist(list("codes" = c(10L, 13L))),
              cdm = cdm,
              ingredientRange = c(2, 10))
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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- subsetOnRouteCategory(
              x = newCodelist(list("codes" = c(20,21))),
              cdm = cdm,
              routeCategory = "topical")
codes
```

`subsetOnVocabulary` *Subset a codelist to only those codes from a particular vocabulary.*

Description

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

Usage

```
subsetOnVocabulary(x, cdm, vocabulary, negate = FALSE)
```

Arguments

`x`	A codelist.
`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`vocabulary`	Vocabulary to subset with (i.e., SNOMED)
`negate`	If FALSE, only concepts with the vocabulary specified will be returned. If TRUE, concepts with the vocabulary specified will be excluded.

Value

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

Examples

```
library(CodelistGenerator)
library(omopgenerics)
cdm <- mockVocabRef()
codes <- subsetOnVocabulary(
              x = newCodelist(list("codes" = c(1L,13L,15L))),
              cdm = cdm,
              vocabulary = "SNOMED")
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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
`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

```
library(CodelistGenerator)
library(omopgenerics)
cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
                           keywords = "arthritis",
                           domains = "Condition",
                           includeDescendants = FALSE)
x <- subsetToCodesInUse(newCodelist(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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
countBy	Either "record" for record-level counts or "person" for person-level counts.

Value

A tibble with summarised counts.

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef("database")
oa <- getCandidateCodes(cdm = cdm, keywords = "osteoarthritis")
codelist <- omopgenerics::newCodelist(list(oa = oa$concept_id))
result_achilles <- summariseAchillesCodeUse(codelist, 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)),
  useSourceCodes = FALSE
)
```

Arguments

x	A codelist.
cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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.
useSourceCodes	Whether the codelist provided contains source codes (TRUE) or standard codes (FALSE).

Value

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

Examples

```
## Not run:
library(omopgenerics)
library(CodelistGenerator)
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 <- newCodelist(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(
  cdm,
  cohortTable,
  x = NULL,
  cohortId = NULL,
  timing = "any",
  countBy = c("record", "person"),
  byConcept = TRUE,
  byYear = FALSE,
  bySex = FALSE,
  ageGroup = NULL,
  useSourceCodes = FALSE
)
```

Arguments

cdm	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
cohortTable	A cohort table from the cdm reference.
x	A codelist or cohort table name.
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.
useSourceCodes	Whether the codelist provided contains source codes (TRUE) or standard codes (FALSE).

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(omopgenerics::newCodelist(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 to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).
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 availableDomains(). 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

```
library(CodelistGenerator)

cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
                           keywords = "Musculoskeletal disorder",
                           domains = "Condition",
                           includeDescendants = FALSE)
codelist <- omopgenerics::newCodelist(list("msk" = codes$concept_id))
orphan_codes <- summariseOrphanCodes(x = codelist,
                                     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"),
  groupColumn = character(),
  hide = character(),
  style = "default",
  .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.
`style`	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: (1) Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See visOmopResults::tableStyle() for available options. (2) Custome Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See visOmopResults::tableStyleCode() for structural examples.
`.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

```
library(CodelistGenerator)
library(omopgenerics)

cdm <- mockVocabRef("database")
oa <- getCandidateCodes(cdm = cdm, keywords = "osteoarthritis")
result_achilles <- summariseAchillesCodeUse(newCodelist(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 = c("date_range_start", "date_range_end"),
  style = NULL,
  .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.
style	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: (1) Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See visOmopResults::tableStyle() for available options. (2) Custome Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See visOmopResults::tableStyleCode() for structural examples.
.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:
library(omopgenerics)
library(CodelistGenerator)
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(newCodelist(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(),
  style = NULL
)
```

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.
`style`	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: (1) Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See visOmopResults::tableStyle() for available options. (2) Custome Code (Advanced): Supply

a block of custom R code. This code must be specific to the table type. See visOmopResults::tableStyleCode() for structural examples.

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(),
```

```
  style = NULL,
  .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.
style	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: (1) Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See visOmopResults::tableStyle() for available options. (2) Custome Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See visOmopResults::tableStyleCode() for structural examples.
.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

```
library(CodelistGenerator)
library(omopgenerics)
cdm <- mockVocabRef("database")
codes <- getCandidateCodes(cdm = cdm,
keywords = "Musculoskeletal disorder",
domains = "Condition",
includeDescendants = FALSE)

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

```
tableOrphanCodes(orphan_codes)

CDMConnector::cdmDisconnect(cdm)
```

`unionCodelists` *Generate a codelist from the union of different codelists. The generated codelist will come out in alphabetical order.*

Description

Generate a codelist from the union of different codelists. The generated codelist will come out in alphabetical order.

Usage

```
unionCodelists(x, keepOriginal = FALSE)
```

Arguments

`x`	A codelist.
`keepOriginal`	Whether to keep the original codelist (TRUE) or just return the stratified ones (FALSE).

Value

A codelist

Examples

```
library(CodelistGenerator)
cdm <- mockVocabRef()
getDrugIngredientCodes(cdm,
                       nameStyle = "{concept_name}") |>
unionCodelists()
```

`vocabularyVersion` *Get the available version of the vocabulary used in the cdm*

Description

Get the available version of the vocabulary used in the cdm

Usage

```
vocabularyVersion(cdm)
```

Arguments

`cdm`	A cdm reference to an OMOP CDM dataset. If data is held within a database, the vocabulary tables should be in the same schema as the clinical tables (person, observation period, and so on).

Value

The vocabulary version being used in the cdm.

Examples

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

Index