

# Package ‘Rdiagnosislist’

July 21, 2025

**Title** Manipulate SNOMED CT Diagnosis Lists

**Version** 1.4.0

**Description** Functions and methods for manipulating 'SNOMED CT' concepts. The package contains functions for loading the 'SNOMED CT' release into a convenient R environment, selecting 'SNOMED CT' concepts using regular expressions, and navigating the 'SNOMED CT' ontology. It provides the 'SNOMEDconcept' S3 class for a vector of 'SNOMED CT' concepts (stored as 64-bit integers) and the 'SNOMEDcodelist' S3 class for a table of concepts IDs with descriptions. The package can be used to construct sets of 'SNOMED CT' concepts for research (<[doi:10.1093/jamia/ocac158](https://doi.org/10.1093/jamia/ocac158)>). For more information about 'SNOMED CT' visit <<https://www.snomed.org/>>.

**License** GPL-3

**Imports** data.table (>= 1.12.8), bit64, methods

**Depends** readxl, R (>= 3.5.0)

**Suggests** knitr, rmarkdown, testthat

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**Author** Anoop D. Shah [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-8907-5724>>)

**Maintainer** Anoop D. Shah <[anoop@doctors.org.uk](mailto:anoop@doctors.org.uk)>

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2025-01-29 08:30:07 UTC

## Contents

acronyms .....	3
----------------	---

addComposeLookupToCDB . . . . .	4
addInactiveConcepts . . . . .	5
addWordnet . . . . .	5
as.data.frame.SNOMEDconcept . . . . .	7
attrConcept . . . . .	8
batchDecompose . . . . .	9
c.SNOMEDconcept . . . . .	10
compose . . . . .	10
createCDB . . . . .	12
createSNOMEDindices . . . . .	13
createTransitive . . . . .	14
decompose . . . . .	14
description . . . . .	16
downloadOrphanet . . . . .	17
downloadWordnet . . . . .	18
exclude_irrelevant_findings . . . . .	19
expandSNOMED . . . . .	20
export . . . . .	21
exportMiADECDB . . . . .	22
exportSNOMEDenvir . . . . .	24
getMaps . . . . .	24
getRefset . . . . .	26
getSNOMED . . . . .	27
hasAttributes . . . . .	28
HISTORY . . . . .	29
htmlCodelistHierarchy . . . . .	30
inactiveIncluded . . . . .	31
is.SNOMEDcodelist . . . . .	32
is.SNOMEDconcept . . . . .	33
loadREADMAPS . . . . .	33
loadSNOMED . . . . .	35
MANUAL_SYNONYMS . . . . .	36
parents . . . . .	37
print.SNOMEDcodelist . . . . .	38
print.SNOMEDconcept . . . . .	39
print.SNOMEDfindings . . . . .	39
QUERY . . . . .	40
READMAPS . . . . .	41
relatedConcepts . . . . .	42
sampleSNOMED . . . . .	43
semanticType . . . . .	43
simplify . . . . .	44
SNOMEDcodelist . . . . .	45
SNOMED_CONCEPT . . . . .	47
SNOMED_DESCRIPTION . . . . .	48
SNOMED_EXTENDEDMAP . . . . .	49
SNOMED_REFSET . . . . .	50
SNOMED_RELATIONSHIP . . . . .	51

<i>acronyms</i>	3
SNOMED_SIMPLEMAP . . . . .	52
union.SNOMEDconcept . . . . .	53
unique.SNOMEDconcept . . . . .	55
<b>Index</b>	<b>56</b>

---

<i>acronyms</i>	<i>Extract acronyms stated in the description of SNOMED CT concepts</i>
-----------------	---

---

## Description

Returns acronyms, if any, expressed within SNOMED CT descriptions in the form 'ABCD - Another Bland Cardiovascular Disease'.

## Usage

```
acronyms(conceptIds, SNOMED = getSNOMED())
```

## Arguments

<code>conceptIds</code>	character or integer64 vector
<code>SNOMED</code>	environment containing SNOMED dictionary. Defaults to an object named 'SNOMED' in the global environment

## Value

a data.table with the following columns: id, conceptId, type = 'Acronym', term = acronym

## Examples

```
SNOMED <- sampleSNOMED()
acronyms('Heart failure')

# Get all synonyms and acronyms
rva <- SNOMEDconcept('Right ventricular abnormality')
rbind(description(rva, include_synonyms = TRUE), acronyms(rva))
```

---

addComposeLookupToCDB *Add composition lookups to CDB*

---

### Description

Creates a composition lookup table for a set of SNOMED CT concepts based on output of ‘decompose’, and adds it to the CDB

### Usage

```
addComposeLookupToCDB(
  decompositions,
  CDB,
  maxcol = 10,
  SNOMED = getSNOMED(),
  ...
)
```

### Arguments

decompositions	vector of filenames of decompose output (read by fread) or data.frame containing outputs of decompose function
CDB	concept database environment, containing a table called FINDINGS
maxcol	maximum number of attributes columns. If NULL it is determined from the data. It might be helpful to specify it so that downstream databases and programs know exactly how many columns to expect. We suggest setting it to 10 which should handle all possible SNOMED CT concept decompositions.
SNOMED	environment containing a SNOMED CT dictionary
...	other arguments to pass to fread

### Value

CDB environment with an additional data.table ‘COMPOSELOOKUP’ with columns rootId, attr\_1 ... attr\_X (up to maxcol), with, due\_to, without, origId (all with data type integer64 and class ‘SNOMEDconcept’)

### See Also

decompose, compose, batchDecompose

Other CDB functions: [addWordnet\(\)](#), [batchDecompose\(\)](#), [createCDB\(\)](#), [downloadOrphanet\(\)](#), [downloadWordnet\(\)](#)

---

addInactiveConcepts     *Add inactive concepts to a SNOMEDcodelist or SNOMEDconcept vector*

---

### Description

Adds SNOMED concepts linked by the QUERY or HISTORY tables that are mapped to or descendants of concepts in a SNOMEDcodelist or a SNOMEDconcept vector. If a SNOMEDcodelist, it is automatically converted to the 'simple' format (all items enumerated).

### Usage

```
addInactiveConcepts(x, provenance = 0:3, SNOMED = getSNOMED())
```

### Arguments

x	SNOMEDcodelist or SNOMEDconcept object
provenance	vector of provenance values to use
SNOMED	SNOMED environment containing HISTORY and QUERY tables

### Details

It is recommended to use this function to convert a reference into a codelist for running a query against an electronic health record database which might contain historic SNOMED CT concepts.

### Value

SNOMEDcodelist or SNOMEDconcept with linked inactive concepts included

### See Also

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [expandSNOMED\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

---

addWordnet                     *Use WordNet to assist concept database creation*

---

### Description

Adds terms from a WordNet thesaurus to a concept database, matching on term. It is recommended to restrict the wordnet categories to ensure that words with multiple meanings are not linked to the wrong synonym. This function also corrects some known errors in WordNet to avoid them being passed on to the CDB; currently this applies to 'allergy = allergic reaction', 'cuneiform bone = triquetral' and 'trauma' = 'injury', but more corrections can be done if needed.

**Usage**

```
addWordnet(
  CDB_TABLE,
  wn_categories,
  WN,
  CHECK_TABLE = NULL,
  errors_to_remove = list(c("allergy", "allergic reaction"), c("allergic",
    "allergic reaction"), c("trauma", "injury"), c("traumatic", "injury"), c("skinny",
    "skin"), c("cuneiform bone", "triquetral bone"), c("upset", "disorder"),
    c("disorderliness", "disorder")),
  noisy = TRUE
)
```

**Arguments**

CDB_TABLE	data.frame or data.table with columns conceptId (integer64) and term (character, with space before and after) containing existing descriptions in the CDB
wn_categories	WordNet categories to use
WN	WordNet data.table as returned by downloadWordnet
CHECK_TABLE	other table in the same format as CDB_TABLE to check for WordNet synonyms that link to another unrelated concept, where this synonym will be excluded because of the risk of errors
errors_to_remove	list of character vectors of length two containing synonym pairs to be removed. The first entry of the pair will be removed from the WordNet file before it is used for adding to CDB
noisy	whether to output test comments

**Value**

CDB\_TABLE with extra rows for Wordnet synonyms

**References**

<https://wordnet.princeton.edu/>

**See Also**

downloadWordnet

Other CDB functions: [addComposeLookupToCDB\(\)](#), [batchDecompose\(\)](#), [createCDB\(\)](#), [downloadOrphanet\(\)](#), [downloadWordnet\(\)](#)

**Examples**

```
WORDNET <- data.table::data.table(cat = c('noun.body', 'noun.state'),
  wordnetId = bit64::as.integer64('1', '2'),
  synonyms = list(c('heart', 'pump', 'ticker'),
    c('infection', 'infectious')),
```

```

parents = list('cardiovascular system',
              'pathologic process'),
adj = list('cardiac', 'infectious'))
# Add Wordnet synonyms to a concept database table
SNOMED <- sampleSNOMED()
CDB_TABLE <- description(c('Heart', 'Infection'),
                        include_synonyms = TRUE)[type == 'Synonym',
                        list(conceptId, term = paste0(' ', tolower(term), ' '))]
addWordnet(CDB_TABLE, 'noun.state', WORDNET)

```

---

as.data.frame.SNOMEDconcept

*Return the SNOMED CT concept IDs for a set of terms*

---

## Description

Carries out an exact or regular expression match to return the concept ID for a set of search terms, or converts a character, integer or integer64 vector to a SNOMEDconcept object.

## Usage

```
## S3 method for class 'SNOMEDconcept'
as.data.frame(x, ...)
```

```
## S3 method for class 'SNOMEDconcept'
as.integer64(x, ...)
```

```
SNOMEDconcept(
  x,
  active_only = TRUE,
  exact_match = TRUE,
  unique = TRUE,
  SNOMED = getSNOMED()
)
```

```
as.SNOMEDconcept(x, ...)
```

## Arguments

x	character vector of terms to match, or character vector containing SNOMED CT concept IDs, or 64-bit integer vector containing SNOMED CT concept IDs
...	additional arguments to send to grepl if using regular expression matching
active_only	whether or not to include inactive concepts
exact_match	if TRUE, only an exact (case sensitive) match is performed. If FALSE, a regular expression match is performed.
unique	whether to include no more than one instance of each SNOMED CT concept
SNOMED	environment containing SNOMED dictionary. Defaults to an object named 'SNOMED' in the global environment

**Value**

a SNOMEDconcept object (vector of 64-bit integers) containing unique SNOMED CT concept IDs

**See Also**

Other SNOMEDconcept functions: [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

**Examples**

```
SNOMEDconcept('Heart failure', SNOMED = sampleSNOMED()) -> hf
is.SNOMEDconcept(hf)
SNOMEDconcept('900000000000003001')
as.SNOMEDconcept('900000000000003001')
```

---

attrConcept

*Retrieve all attributes of a set of SNOMED CT concepts*


---

**Description**

Returns the portion of the SNOMED CT relationship tables containing relationships where the given concepts are either the source or the destination.

**Usage**

```
attrConcept(
  conceptIds,
  SNOMED = getSNOMED(),
  tables = c("RELATIONSHIP", "STATEDRELATIONSHIP"),
  active_only = TRUE
)
```

**Arguments**

conceptIds	character or integer64 vector of SNOMED concept IDs
SNOMED	environment containing a SNOMED dictionary
tables	character vector of relationship tables to use
active_only	whether to return only active attributes



**Value**

a data.table with the following columns: sourceId (concept ID of source for relationship), destinationId (concept ID of source for relationship), typeId (concept ID of relationship type), typeName (description of relationship type)

**Examples**

```
SNOMED <- sampleSNOMED()

attrConcept(as.SNOMEDconcept('Heart failure'))
```

---

batchDecompose	<i>Creates a set of lookups for SNOMED composition</i>
----------------	--

---

**Description**

Creates composition lookup table for a set of SNOMED CT concepts and exports the results to a file

**Usage**

```
batchDecompose(conceptIds, CDB, output_filename, SNOMED = getSNOMED(), ...)
```

**Arguments**

conceptIds	SNOMED CT concept IDs for creating decompositions
CDB	concept database environment, containing a table called FINDINGS
output_filename	filename of output file
SNOMED	environment containing a SNOMED dictionary
...	other arguments to pass through to decompose

**Value**

TRUE if successful

**See Also**

decompose, compose, addComposeLookupToCDB

Other CDB functions: [addComposeLookupToCDB\(\)](#), [addWordnet\(\)](#), [createCDB\(\)](#), [downloadOrphanet\(\)](#), [downloadWordnet\(\)](#)

---

c.SNOMEDconcept	<i>Concatenate vectors of SNOMED CT concepts</i>
-----------------	--

---

### Description

SNOMEDconcept is an S3 class for vectors of SNOMED concept IDs as 64-bit integers. This function concatenates two or more SNOMEDconcept vectors.

### Usage

```
## S3 method for class 'SNOMEDconcept'  
c(...)
```

### Arguments

... SNOMEDconcept vectors

### Value

concatenation of vectors

### See Also

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

### Examples

```
hf <- SNOMEDconcept('Heart failure', SNOMED = sampleSNOMED())  
hf2 <- c(hf, hf)
```

---

compose	<i>Select more specific SNOMED CT concepts based on attributes</i>
---------	--

---

### Description

Finds the most specific SNOMED CT concepts that matches the combination of a root concept and attributes. Based on a composeLookup table

**Usage**

```
compose(
  conceptId,
  CDB,
  attributes_conceptIds = bit64::integer64(0),
  due_to_conceptIds = bit64::integer64(0),
  without_conceptIds = bit64::integer64(0),
  with_conceptIds = bit64::integer64(0),
  SNOMED = getSNOMED()
)
```

**Arguments**

conceptId	SNOMED CT concept to refine
CDB	SNOMED CT concept database, as created by createCDB. An environment containing the following data tables: FINDINGS, QUAL, CAUSES, BODY, FINDINGS, OTHERSUB, OVERLAP, TRANSITIVE. Must also contain a COMPOSELOOKUP table created by addComposeLookupToCDB
attributes_conceptIds	SNOMED concept Ids of attributes of concept e.g. laterality, severity, acuteness
due_to_conceptIds	SNOMED concept Ids of cause
without_conceptIds	SNOMED concept Ids of conditions stated to be absent
with_conceptIds	SNOMED concept Ids of conditions also present
SNOMED	environment containing SNOMED CT tables

**Value**

a refined SNOMED concept Id

**See Also**

decompose, batchDecompose, addComposeLookupToCDB

**Examples**

```
## Not run:
SNOMED <- sampleSNOMED()
miniCDB <- createCDB(SNOMED)

D <- decompose('Cor pulmonale', CDB = miniCDB)
print(D)

# -----
# 83291003 | Cor pulmonale (disorder)
# -----
```

```

# Root : 367363000 | Right ventricular failure (disorder)
# - Due to : 19829001 | Disorder of lung (disorder)

# Compile decompositions into a lookup table
miniCDB <- addComposeLookupToCDB(D, CDB = miniCDB)

compose(as.SNOMEDconcept('Right heart failure'),
  due_to_conceptIds = as.SNOMEDconcept('Disorder of lung'),
  CDB = miniCDB, composeLookup = CL)
# [1] "83291003 | Cor pulmonale (disorder)"

## End(Not run)

```

---

createCDB

*Creates an environment containing CDB files*


---

## Description

Extracts SNOMED CT concepts from appropriate places in the hierarchy to create a set of CDB files in an environment. Uses WordNet and manual synonyms if available.

## Usage

```

createCDB(
  SNOMED = getSNOMED(),
  TRANSITIVE = NULL,
  WN = NULL,
  MANUAL_SYNONYMS = NULL,
  noisy = TRUE,
  stopwords = c("the", "of", "by", "with", "to", "into", "and", "or", "at", "as",
    "and/or", "in")
)

```

## Arguments

SNOMED	environment containing a SNOMED dictionary
TRANSITIVE	transitive closure table, generated by createTransitive. It is regenerated if not provided.
WN	WordNet data.table as returned by downloadWordnet containing WordNet data from appropriate categories, in the format: cat (character), wordnetId (integer64), synonyms (list), parents (list), adj (list)
MANUAL_SYNONYMS	data.table with columns term1 and term2, containing additional exact synonyms or abbreviations
noisy	whether to output status messages
stopwords	vector of stopwords

**Value**

environment containing the following data tables: FINDINGS, QUAL, CAUSES, BODY, OTHER-CAUSE, OTHERSEARCH, OVERLAP, TRANSITIVE

**See Also**

exportMiADECDB, MANUAL\_SYNONYMS

Other CDB functions: [addComposeLookupToCDB\(\)](#), [addWordnet\(\)](#), [batchDecompose\(\)](#), [downloadOrphanet\(\)](#), [downloadWordnet\(\)](#)

**Examples**

```
# Not run
# data(MANUAL_SYNONYMS)
# WN <- downloadWordnet()
# MANUAL_SYNONYMS <- rbind(MANUAL_SYNONYMS, downloadOrphanet())
# CDBNEW <- createCDB(WN = WN, MANUAL_SYNONYMS = MANUAL_SYNONYMS)
```

---

createSNOMEDindices    *Create indices for tables in a SNOMED environment*

---

**Description**

Creates relevant indices for fast searching of SNOMED CT tables

**Usage**

```
createSNOMEDindices(SNOMED)
```

**Arguments**

SNOMED            environment containing data.table objects: CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP

**Value**

The environment with indices added to each table for fast searching

**See Also**

CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP, REFSET, SIMPLEMAP, EXTENDEDMAP, loadSNOMED, sampleSNOMED

---

createTransitive	<i>Create a transitive closure table for is-a relationships for faster ancestor / descendant lookups</i>
------------------	--

---

### Description

Returns a data.table containing ancestor / descendant relationships which can be used in ancestors and descendants functions

### Usage

```
createTransitive(
  conceptIds,
  SNOMED = getSNOMED(),
  tables = c("RELATIONSHIP", "STATEDRELATIONSHIP")
)
```

### Arguments

conceptIds	character or integer64 vector of SNOMED concept IDs for the subset of concepts to include in the transitive table.
SNOMED	environment containing a SNOMED dictionary
tables	vector of names of relationship table(s) to use; by default use both RELATIONSHIP and STATEDRELATIONSHIP

### See Also

[ancestors()] and [descendants()]

### Examples

```
SNOMED <- sampleSNOMED()
TRANSITIVE <- createTransitive('Heart failure')
```

---

decompose	<i>Decomposition of meaning of a finding or disorder SNOMED CT concept</i>
-----------	--

---

### Description

Decomposes a SNOMED CT term into separate components according to the SNOMED CT information model and text parsing. Each term may have a number of possible decompositions. Requires a CDB environment created by createCDB.

**Usage**

```
decompose(
  conceptIds,
  diagnosis_text = NULL,
  CDB,
  SNOMED = getSNOMED(),
  noisy = FALSE,
  omit_unmatched = TRUE
)
```

**Arguments**

<code>conceptIds</code>	vector of SNOMED CT concepts to decompose
<code>diagnosis_text</code>	vector of SNOMED CT terms (or in theory any text that has the same meaning as the SNOMED CT concept). If <code>NULL</code> , decompositions are created for all SNOMED CT synonyms of the concepts.
<code>CDB</code>	an environment containing CDB files, as created by <code>createCDB</code>
<code>SNOMED</code>	an environment containing the SNOMED CT dictionary
<code>noisy</code>	whether to output messages (for debugging)
<code>omit_unmatched</code>	whether to omit rows in which some attributes could not be matched to SNOMED CT concepts

**Value**

a `SNOMEDfinding` objects, which is a `data.table` with columns `rootId` (`integer64`), `with` (`integer64`), `due_to` (`integer64`), `after` (`integer64`), `without` (`integer64`), `body_site` (`integer64`), `severity` (`integer64`), `stage` (`integer64`), `laterality` (`integer64`), `roottext` (`character`), `partId` (`integer64`), `other_conceptId` (`character`, a string of SNOMED CT concept IDs separated by spaces), `origId` (`integer64`)

**Examples**

```
## Not run:
miniCDB <- createCDB(SNOMED = sampleSNOMED())
D <- decompose(as.SNOMEDconcept('Cor pulmonale',
  SNOMED = sampleSNOMED()), CDB = miniCDB, SNOMED = sampleSNOMED())

## End(Not run)

# -----
# 83291003 | Cor pulmonale (disorder)
# -----
# Root : 367363000 | Right ventricular failure (disorder)
# - Due to : 19829001 | Disorder of lung (disorder)
```

---

description	<i>Obtain descriptions for a set of SNOMED CT terms</i>
-------------	---

---

## Description

Returns the descriptions matching a set of concept IDs from a SNOMED dictionary

## Usage

```
description(  
  conceptIds,  
  include_synonyms = FALSE,  
  active_only = TRUE,  
  SNOMED = getSNOMED()  
)
```

## Arguments

conceptIds	character or integer64 vector
include_synonyms	whether to return all synonyms, or just the Fully Specified Name, ensuring just one row per concept (default)
active_only	whether to include only active descriptions
SNOMED	environment containing SNOMED dictionary. Defaults to an object named 'SNOMED' in the global environment

## Value

a data.table with the following columns: id, conceptId, type (only if include\_synonyms = TRUE), term, active (only if active\_only = FALSE)

## See Also

`htmlCodelistHierarchy`

## Examples

```
hf <- SNOMEDconcept('Heart failure', SNOMED = sampleSNOMED())  
description(hf, include_synonyms = FALSE, SNOMED = sampleSNOMED())
```



---

downloadOrphanet	<i>Download Orphanet to assist with adding synonyms for rare diseases</i>
------------------	---

---

### Description

Downloads the Orphanet nomenclature pack and converts it into a format to be appended to MANUAL\_SYNONYMS and used in concept database creation.

### Usage

```
downloadOrphanet(  
  orphanet_url =  
    "https://www.orphadata.com/data/nomenclature/packs/Orphanet_Nomenclature_Pack_EN.zip",  
  masterfile_name = NULL,  
  SNOMED = getSNOMED()  
)
```

### Arguments

orphanet_url	URL or filepath to Orphanet zip file
masterfile_name	name of xls file containing Orphanet synonyms. If omitted, it is assumed to be the only file with 'MasterFile' in its name, and the program will search for it in the Orphanet zip file.
SNOMED	environment containing SNOMED CT dictionary

### Value

data.table containing Orphanet synonyms in the format

### References

<https://www.orphadata.com/pack-nomenclature/>

### See Also

downloadWordnet, MANUAL\_SYNONYMS

Other CDB functions: [addComposeLookupToCDB\(\)](#), [addWordnet\(\)](#), [batchDecompose\(\)](#), [createCDB\(\)](#), [downloadWordnet\(\)](#)

### Examples

```
# Not run  
# ORPHANET <- downloadOrphanet()
```

---

downloadWordnet	<i>Download WordNet to assist concept database creation</i>
-----------------	---

---

### Description

Downloads the WordNet thesaurus and converts it into a format to be used by addWordnet to add extra synonyms to a concept database.

### Usage

```
downloadWordnet(  
  wordnet_url = "https://wordnetcode.princeton.edu/wn3.1.dict.tar.gz",  
  wn_categories = c("noun.body", "noun.state", "noun.process", "noun.animal",  
    "noun.plant", "noun.phenomenon")  
)
```

### Arguments

wordnet\_url     URL or filepath to WordNet tar.gz file  
wn\_categories    WordNet categories from which to extract data

### Value

data.table containing WordNet data from appropriate categories, in the format: cat (character), wordnetId (integer64), synonyms (list), parents (list), adj (list)

### References

<https://wordnet.princeton.edu/>

### See Also

addWordnet

Other CDB functions: [addComposeLookupToCDB\(\)](#), [addWordnet\(\)](#), [batchDecompose\(\)](#), [createCDB\(\)](#), [downloadOrphanet\(\)](#)

### Examples

```
# Not run  
# WORDNET <- downloadWordnet()
```

---

`exclude_irrelevant_findings`*Sample inclusion, exclusion and blacklist sets for a MiADE CDB*

---

### Description

Returns a set of SNOMED concepts (as a SNOMEDconcept vector) which can be used to exclude findings in the MedCAT named entity recognition step, or blacklist (filter out) findings from the final output.

### Usage

```
exclude_irrelevant_findings(SNOMED = getSNOMED())
```

```
blacklist_vague_findings(SNOMED = getSNOMED())
```

```
blacklist_almost_all_except_diseases(SNOMED = getSNOMED())
```

### Arguments

SNOMED                    environment containing a SNOMED dictionary

### Details

**exclude\_irrelevant\_findings** social history (except housing problems and care needs), administrative statuses (except registered disabled) and for concept detection

**blacklist\_vague\_findings** vague findings and disorders, intended to be used in the blacklist

**blacklist\_almost\_all\_except\_diseases** almost all findings and vague disorders, intended to be used in the blacklist

### Value

SNOMEDconcept vector containing findings to exclude

### See Also

`exportMiADECDB`, `createCDB`

Other MiADE functions: [exportMiADECDB\(\)](#)

Other MiADE functions: [exportMiADECDB\(\)](#)

Other MiADE functions: [exportMiADECDB\(\)](#)

---

expandSNOMED                      *Expand or contract a SNOMEDcodelist*

---

### Description

SNOMEDcodelist is an S3 class for sets of SNOMED concepts. In the 'contracted' form, it may contain only parents and not child terms (to create a more succinct list). The 'Expanded' form contains all concepts. The output of 'showCodelistHierarchy' includes all hierarchies contained within the codelist in a format suitable for display.

### Usage

```
expandSNOMED(x, SNOMED = getSNOMED(), ...)

contractSNOMED(x, SNOMED = getSNOMED(), ...)

showCodelistHierarchy(
  x,
  SNOMED = getSNOMED(),
  max_excluded_descendants = 200,
  ...
)
```

### Arguments

x	SNOMEDcodelist to expand or contract. If x is not a SNOMEDcodelist, it is coerced to one by <code>as.SNOMEDcodelist</code>
SNOMED	environment containing a SNOMED dictionary
...	other arguments to pass to <code>as.SNOMEDcodelist</code>
max_excluded_descendants	(integer) whether to show excluded descendants as long as they do not exceed this number (a limit is suggested to avoid the program crashing if there are too many descendants). If this number is exceeded, the program will initially try to include children only, and if there are still too many, it will ignore all descendants. An 'included' column is added to the codelist showing which terms are included. This can make it easy to see if a codelist is consistent with the SNOMED CT ontology.

### Value

An object of class 'SNOMEDcodelist' with attribute `Expanded = TRUE`

### See Also

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

## Examples

```
SNOMED <- sampleSNOMED()

my_concepts <- SNOMEDconcept('Heart failure')
my_codelist <- SNOMEDcodelist(data.frame(conceptId = my_concepts,
  include_desc = TRUE))
expanded_codelist <- expandSNOMED(my_codelist)
contractSNOMED(expanded_codelist)
```

---

export

*Export a SNOMEDcodelist*

---

## Description

Writes a SNOMEDcodelist to file. If the filename is NULL, a filename is created from the 'codelist\_name' attribute.

## Usage

```
export(x, ...)

## S3 method for class 'SNOMEDcodelist'
export(x, filename = NULL, metadata = TRUE, ...)
```

## Arguments

x	SNOMEDcodelist object to export to file
...	not used
filename	character vector of length 1 for the file to write to. If NULL, a filename is generated from the codelist filename.
metadata	whether metadata is exported (default = TRUE)

## Value

invisibly returns the exported codelist

## See Also

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [expandSNOMED\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [expandSNOMED\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

---

 exportMiADECDB

*Exports CDB files for MedCAT / MiADE*


---

### Description

Produces a set of files for the findings / problems algorithm of MedCAT and MiADE. Uses the CDB environment created using createCDB which can incorporate additional manual synonyms or synonyms from WordNet.

### Usage

```
exportMiADECDB(
  CDB,
  export_folderpath,
  lang_refset_files = NULL,
  exclude = NULL,
  include = NULL,
  exclude_historic = descendants("Disorder", SNOMED = getSNOMED()),
  blacklist = NULL,
  SNOMED = getSNOMED()
)
```

### Arguments

CDB	concept database environment created by createCDB
export_folderpath	folder path to export to
lang_refset_files	character vector of file paths to SNOMED CT language refset files, in order to identify the preferred term for each concept. If NULL, the Fully Specified Name minus the semantic type suffix is used as the preferred term (e.g. if the Fully Specified Name is ‘Cancer (disorder)’, the default preferred term is ‘Cancer’.
exclude	a SNOMEDconcept or SNOMEDcodelist object specifying concepts to exclude from the concept database. By default, all concepts in the FINDINGS, CAUSES, BODY, LATERALITY, MORPH, SEVERITY, STAGE and QUAL tables will be included.
include	a SNOMEDconcept or SNOMEDcodelist object specifying additional concepts to include in the concept database. By default, all findings are included for potential export, but there may additional concepts of other semantic types (e.g. situation concepts) that need to be included. Inclusion takes place after exclusion, i.e. a concept in both the include and exclude lists will be included.
exclude_historic	a SNOMEDconcept or SNOMEDcodelist object specifying concepts to be excluded from the ‘historic’ lookup, i.e. those that should not be converted into historic forms. The default is to not do this conversion for disorders, only for procedures.

blacklist	a SNOMEDconcept or SNOMEDcodelist object specifying concepts to filter out of the final output. By default, concepts in the CDB of any semantic type other than 'finding' or 'disorder' are excluded. The blacklist can be used to exclude a subset of findings or disorders that are not useful for the particular application.
SNOMED	environment containing a SNOMED dictionary

### Details

The following files are exported:

For MedCAT (named entity recognition and linking):

**problems\_cdb.csv** - CSV file in MedCAT concept database format containing cui (SNOMED CT concept ID), name, name\_status ('P' for preferred term, 'N' for terms that must be disambiguated (e.g. acronyms or short terms), 'A' for synonym), ontologies = SNO (for SNOMED CT)

For MiADE postprocessing:

**negated.csv** - CSV file with columns findingId (SNOMED CT concept ID of the underlying finding / disorder) and situationId (SNOMED CT concept ID of the pre-coordinated situation concept for negation of the finding / disorder). Sorted by findingId.

**historic.csv** - CSV file with columns findingId (SNOMED CT concept ID of the underlying finding / disorder) and situationId (SNOMED CT concept ID of the pre-coordinated situation concept for 'history of' the finding / disorder). Sorted by findingId.

**suspected.csv** - CSV file with columns findingId (SNOMED CT concept ID of the underlying finding / disorder) and situationId (SNOMED CT concept ID of the pre-coordinated situation concept for 'suspected' finding / disorder). Sorted by findingId.

**overlap.csv** - CSV file with columns findingId (SNOMED CT concept ID of the underlying finding / disorder) and otherId (SNOMED CT concept ID of a concept with the same description but a different semantic type, typically a morphologic abnormality). Sorted by otherId.

**problem\_blacklist.csv** - CSV file without header with one column containing SNOMED CT concept IDs for concepts that may be identified by MedCAT as part of text analysis but should not be included in final MiADE output, Examples include procedure codes which may be used to link to pre-coordinated 'history of...' concepts. This file can also be used to force MiADE to ignore any specific SNOMED CT concepts in the output. Sorted in ascending order.

For more information about MiADE, visit <https://www.ucl.ac.uk/health-informatics/research/miade/miade-software-and-availability>

For more information about MedCAT, visit <https://github.com/CogStack/MedCAT>

### Value

TRUE if successful

### See Also

createCDB, downloadWordnet, downloadOrphanet, MANUAL\_SYNONYMS, exclude\_irrelevant\_findings

Other MiADE functions: [exclude\\_irrelevant\\_findings\(\)](#)

**Examples**

```
# Not run
# exportMiADECDB(CDB, export_folderpath = tempdir())
```

---

```
exportSNOMEDenvir      Export a SNOMED environment to a folder
```

---

**Description**

Creates tab separated files which can be reloaded with relevant indices for fast searching of SNOMED CT tables

**Usage**

```
exportSNOMEDenvir(SNOMED, folder)
```

**Arguments**

SNOMED	environment containing data.table objects: CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP, REFSET, SIMPLEMAP, EXTENDEDMAP
folder	path to folder where files will be written

**See Also**

CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP

---

```
getMaps      Obtain Read 2, CTV3, ICD-10 and OPCS4 maps for SNOMED CT concepts
```

---

**Description**

Returns concepts mapped to SNOMED CT from either the SIMPLEMAP table in the SNOMED dictionary (Clinical Terms Version 3, CTV3 maps, one per concept), the EXTENDEDMAP table (ICD-10 and OPCS4 maps) or a separate mapping table with Read Clinical Terms Version 2 (Read 2) and CTV3 maps. A sample mapping table (READMAPS) is provided.

**Usage**

```
getMaps(
  x,
  mappingtable = NULL,
  to = c("read2", "ctv3", "icd10", "opcs4", "ctv3simple"),
  SNOMED = getSNOMED(),
  single_row_per_concept = TRUE
)
```



**Arguments**

x	SNOMEDcodelist or SNOMEDconcept object. If it is a SNOMEDconcept object it is first converted to a SNOMEDcodelist. If it is a SNOMEDcodelist it is first converted to 'simple' format. Columns named 'read2_code' or 'read2_term' (if adding Read 2 maps) or 'ctv3_concept' or 'ctv3_termid' (if adding CTV3 maps) will be overwritten.
mappingtable	data.table containing mapping in the format described in 'Details'. The MAPS dataset in this package provides a sample. It must contain a unique field 'conceptId', and fields named 'read2_code' and 'read2_term' (for mapping to Read 2) or 'ctv3_concept' and 'ctv3_termid' (for mapping to CTV3).
to	character vector stating which terminologies to map to. Options are 'icd10', 'opcs4', 'ctv3simple' (use tables included within the SNOMED dictionary), or 'read2' or 'ctv3' (require a separate mapping table such as READMAPS). Beware that including multiple destination terminologies may result in a significant expansion of the number of rows if single_row_per_concept is FALSE.
SNOMED	an environment containing the SNOMED CT dictionary. If not supplied, it will be obtained using getSNOMED().
single_row_per_concept	(logical) if TRUE (default), the function returns a single row per concept with Read 2 and CTV3 maps returned as lists (i.e. multiple entries within a single cell). This means the output is a valid SNOMEDcodelist object. If FALSE, returns multiple rows per concept (one for each map).

**Details**

The mapping table can be created from the NHS Digital 'Data Migration' pack files which contain 'forward' maps of Read 2 and CTV3 to SNOMED CT. These are intended for converting individual entries in electronic health records to SNOMED CT. The 'forward' map files contain a SNOMED CT map for every Read 2 or CTV3 code, but not all the SNOMED CT concepts are mapped. Future SNOMED CT concepts will also not be mapped.

These maps can be used for converting SNOMED CT codelists into Read 2 or CTV3 format for running queries, such as to characterise patient phenotypes or identify patient populations for research. They cannot be used in the reverse direction (to map a Read 2/CTV3 codelist to SNOMED CT) because some of the SNOMED CT terms will be missed out, and the list will be incomplete.

The mapping table must be a data.table object with columns: conceptId (integer64, unique), read2\_code (character list of 7-character Read 2 codes), read2\_term (character list of Read 2 terms), ctv3\_concept (character list of CTV3 concept codes), ctv3\_termid (character list of CTV3 term description codes)

**Value**

a data.table containing the columns conceptId and either 'read2\_code' and 'read2\_term' (for mapping to Read 2), 'ctv3\_concept' and 'ctv3\_termid' (for mapping to CTV3 using the mapping table), 'ctv3\_simple' (mapping to CTV3 using SIMPLEMAP within the SNOMED dictionary), 'icd10\_code' or 'opcs4\_code' (mapped using EXTENDEDMAP within the SNOMED dictionary). If single\_row\_per\_concept is TRUE, the mapped rows are of type 'list' and the output is also a SNOMEDcodelist in 'simple' format, otherwise the output may have multiple rows per conceptId.

Note that each Read 2, CTV3, ICD-10 or OPCS4 term may be mapped to multiple SNOMED CT concepts.

### See Also

READMAPS, loadREADMAPS

### Examples

```
# Load sample SNOMED CT dictionary into the global environment
# so it is available to the functions in this example
SNOMED <- sampleSNOMED()
# Use the sample READMAPS table in this package
data(READMAPS)

# Example: Mapping a single concept
getMaps(SNOMEDconcept('Heart failure'), mappingtable = READMAPS,
        to = 'read2')
# Example: Mapping a concept and its descendants
getMaps(descendants(SNOMEDconcept('Heart failure')),
        mappingtable = READMAPS, to = 'read2')
# Example: Mapping a codelist
getMaps(SNOMEDcodelist(SNOMEDconcept('Heart failure')),
        mappingtable = READMAPS, to = c('ctv3', 'ctv3simple', 'icd10'))
```

---

getRefset

*Retrieves a Refset from the REFSET table*

---

### Description

Retrieves a Refset from the REFSET table

### Usage

```
getRefset(conceptIds, SNOMED = getSNOMED())
```

### Arguments

conceptIds	character or integer64 vector of Refset SNOMED concept IDs, or something that can be coerced to a SNOMEDconcept
SNOMED	environment containing a SNOMED dictionary

### Value

a SNOMEDconcept vector of conceptIds of members of the selected refset(s)

## Examples

```
SNOMED <- sampleSNOMED()

getRefset(c('Renal clinical finding simple reference set',
           'Care planning activities simple reference set'))
```

---

getSNOMED	<i>Retrieves SNOMED CT dictionary from the global environment</i>
-----------	---

---

## Description

Returns an object named 'SNOMED' from the global environment. Returns an error if no such object exists, or if it is not an environment containing tables named CONCEPT, RELATIONSHIP, STATEDRELATIONSHIP and DESCRIPTION. There is no attempt to check that these tables are actually valid.

## Usage

```
getSNOMED(SNOMEDname = "SNOMED")
```

## Arguments

SNOMEDname      name of the SNOMED environment to search for

## Value

SNOMED environment from the global environment

## See Also

CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP, REFSET, SIMPLEMAP, EXTENDEDMAP, loadSNOMED, sampleSNOMED

## Examples

```
SNOMED <- sampleSNOMED()
SNOMED2 <- getSNOMED()

# To display metadata for this SNOMED CT dictionary
SNOMED2$metadata
```

---

hasAttributes	<i>Whether SNOMED CT concepts have particular attributes</i>
---------------	--

---

### Description

For each concept in the first list, whether it has the attribute in the second list. Returns a vector of Booleans.

### Usage

```
hasAttributes(
  sourceIds,
  destinationIds,
  typeIdIds = bit64::as.integer64("116680003"),
  SNOMED = getSNOMED(),
  tables = c("RELATIONSHIP", "STATEDRELATIONSHIP"),
  active_only = TRUE
)
```

### Arguments

sourceIds	character or integer64 vector of SNOMED concept IDs for children, recycled if necessary
destinationIds	character or integer64 vector of SNOMED concept IDs for parents, recycled if necessary
typeIds	character or integer64 vector of SNOMED concept IDs for relationship types, recycled if necessary. Defaults to 116680003 = 'Is a' (child/parent)
SNOMED	environment containing a SNOMED dictionary
tables	character vector of relationship tables to use
active_only	whether only active relationships should be considered, default TRUE

### Value

a vector of Booleans stating whether the attribute exists

### Examples

```
data.table::setDTthreads(threads = 1) # for CRAN testing

SNOMED <- sampleSNOMED()

hasAttributes(c('Heart failure', 'Acute heart failure'),
  c('Heart structure', 'Heart failure'),
  c('Finding site', 'Is a'))
```

---

 HISTORY

*Sample history substitution table from SNOMED CT dictionary*


---

**Description**

Sample of the SNOMED CT table showing current equivalents for inactive concepts.

**Usage**

data(HISTORY)

**Format**

An object of class "data.table"

**Details**

**OLDCONCEPTID** integer64: concept ID of the inactive concepts

**OLDCONCEPTSTATUS** integer: status of the old concept

**NEWCONCEPTID** integer64: concept ID of the new concept

**NEWCONCEPTSTATUS** integer: status of the new concept

**PATH** character: path

**ISAMBIGUOUS** integer: whether ambiguous

**ITERATIONS** integer: number of iterations

**OLDCONCEPTFSN** character: old concept Fully Specified Name

**OLDCONCEPTFSN\_TAGCOUNT** integer: number of tags for old concept

**NEWCONCEPTFSN** integer: new concept Fully Specified Name

**NEWCONCEPTFSN\_STATUS** integer: new concept Fully Specified Name status

**TLH\_IDENTICALFLAG** integer: whether TLH identical

**FSN\_TAGLESSIDENTICALFLAG** integer: whether Fully Specified Names are identical ignoring the tags

**FSN\_TAGIDENTICALFLAG** integer: whether Fully Specified Names tags are identical

**See Also**

Other SNOMEDsample: [QUERY](#), [SNOMED\\_RELATIONSHIP](#)

Other sampleSNOMED: [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

**Examples**

```
# Create a TEST environment and load the sample dictionaries
TEST <- new.env()
data(CONCEPT, envir = TEST)
data(DESCRIPTION, envir = TEST)
data(RELATIONSHIP, envir = TEST)
data(STATEDRELATIONSHIP, envir = TEST)
data(HISTORY, envir = TEST)

# Show properties of the history table
str(TEST$HISTORY)
```

---

htmlCodelistHierarchy *Export a SNOMEDcodelist hierarchy to HTML*

---

**Description**

Exports a codelist with hierarchy as HTML for easy viewing.

**Usage**

```
htmlCodelistHierarchy(
  x,
  file = NULL,
  title = NULL,
  description = NULL,
  extracols = NULL,
  SNOMED = getSNOMED(),
  ...
)
```

**Arguments**

x	a SNOMEDcodelist, codelistHierarchy (output of showCodelistHierarchy), or an object which can be coerced to a SNOMEDcodelist (such as a SNOMED-concept vector).
file	filename to export to. If NULL, no file is written
title	title of HTML document
description	paragraph of description text (excluding <p></p> tags)
extracols	character vector of additional columns of codelist_with_hierarchy to include in HTML output
SNOMED	environment containing the SNOMED dictionary to use
...	extra arguments to pass to as.SNOMEDcodelist

**Value**

a character vector containing HTML output

**See Also**

showCodelistHierarchy

**Examples**

```
data.table::setDTthreads(threads = 1) # for CRAN testing
SNOMED <- sampleSNOMED() # use sample SNOMED for testing

my_concepts <- SNOMEDconcept('Acute heart failure')
my_codelist <- SNOMEDcodelist(data.frame(conceptId = my_concepts,
  include_desc = TRUE))
my_codelist <- getMaps(my_codelist, to = 'icd10')
htmlCodelistHierarchy(my_codelist, file = paste0(tempdir(),
  '/codelist.html'), extracols = 'icd10_code')
# The codelist.html file can now be viewed in a web browser

# Clean up temporary file
file.remove(paste0(tempdir(), '/codelist.html'))
```

---

inactiveIncluded	<i>Check if inactive terms are included in SNOMED CT dictionary</i>
------------------	---

---

**Description**

Checks the active\_only flag in the metadata of a SNOMED environment to determine whether inactive terms are included

**Usage**

```
inactiveIncluded(SNOMED = getSNOMED())
```

**Arguments**

SNOMED	environment containing SNOMED dictionary, defaults to an object named 'SNOMED' in the global environment
--------	--

**Value**

TRUE or FALSE (logical vector of length one)

**Examples**

```
# Create a TEST environment and load the sample dictionaries
TEST <- sampleSNOMED()
inactiveIncluded(TEST)
assign('metadata', list(active_only = TRUE), envir = TEST)
inactiveIncluded(TEST)
```

---

is.SNOMEDcodelist      *Check if an object is a SNOMEDcodelist*

---

### Description

SNOMEDcodelist is an S3 class for lists of SNOMED codes. This function checks whether the object has the class SNOMEDcodelist, and whether the specified attributes are as per the arguments (if the arguments are left as NULL, as per default, they are not checked). The function does not check if the codelist contains valid data.

### Usage

```
is.SNOMEDcodelist(
  x,
  format = NULL,
  codelist_name = NULL,
  version = NULL,
  author = NULL,
  date = NULL,
  SNOMED = NULL
)
```

### Arguments

x	object to check
format	Whether the codelist is expressed as a simple enumeration of concepts ('simple'), as a set of concept hierarchies ('tree') or as a set of hierarchies showing all concepts ('exptree'). Codelists can be converted between the formats, but the result of conversion may depend on the SNOMED CT dictionary being used.
codelist_name	Name of the codelist (character vector of length 1)
version	Version of the codelist (character vector of length 1)
author	Author of the codelist (character vector of length 1)
date	Date assigned to the codelist (character vector of length 1)
SNOMED	Dummy argument to ensure that this function works with as.SNOMEDcodelist

### Value

a logical vector of length one: TRUE or FALSE

### See Also

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [expandSNOMED\(\)](#), [export\(\)](#), [print.SNOMEDcodelist\(\)](#)



---

is.SNOMEDconcept	<i>Check if an object is a SNOMEDconcept</i>
------------------	--

---

### Description

SNOMEDconcept is an S3 class for vectors of SNOMED concept IDs as 64-bit integers. This function checks whether the object has the class SNOMEDconcept and is a vector of 64-bit integers.

### Usage

```
is.SNOMEDconcept(x)
```

### Arguments

x                    object to check

### Value

a logical vector of length one: TRUE or FALSE

### See Also

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

---

loadREADMAPS	<i>Load mappings from Read to SNOMED CT into an R data.table</i>
--------------	--

---

### Description

Creates a mapping table derived from NHS Digital Data Migration distribution. These tables are available from the Technology Reference Update Distribution: <https://isd.digital.nhs.uk/trud/user/guest/group/0/pack/9/subpack/9/releases>

### Usage

```
loadREADMAPS(  
  not_assured_rcsctmap_uk,  
  not_assured_rctermmap_uk,  
  assured_ctv3sctmap2_uk  
)
```

**Arguments**

**not\_assured\_rcsctmap\_uk**  
File containing Read 2 codes mapped to SNOMED CT, in file: 'Not Clinically Assured/rcsctmap\_uk\_20200401000001.txt'

**not\_assured\_rctermstmap\_uk**  
File containing Read 2 terms mapped to SNOMED CT, in file: 'Not Clinically Assured/rctermstmap\_uk\_20200401000001.txt'

**assured\_ctv3sctmap2\_uk**  
File containing CTV3 concepts and terms mapped to SNOMED CT, in file: 'Clinically Assured/ctv3sctmap2\_uk\_20200401000001.txt'

**Details**

The final release was in April 2020. The mapping tables are intended for converting entries in clinical records from Read Version 2 (Read 2) to SNOMED CT, and Clinical Terms Version 3 (CTV3) to SNOMED CT.

These maps can be used for converting SNOMED CT codelists into Read 2 or CTV3 format for running queries, such as to characterise patient phenotypes or identify patient populations for research. They cannot be used in the reverse direction (to map a Read 2/CTV3 codelist to SNOMED CT) because some of the SNOMED CT terms will be missed out, and the list will be incomplete.

This function uses the following three mapping files:

**not\_assured\_rcsctmap\_uk** File containing Read 2 codes mapped to SNOMED CT, in file: 'Not Clinically Assured/rcsctmap\_uk\_20200401000001.txt'

**not\_assured\_rctermstmap\_uk** File containing Read 2 terms mapped to SNOMED CT, in file: 'Not Clinically Assured/rctermstmap\_uk\_20200401000001.txt'

**assured\_ctv3sctmap2\_uk** File containing CTV3 concepts and terms mapped to SNOMED CT, in file: 'Clinically Assured/ctv3sctmap2\_uk\_20200401000001.txt'

The output data.table has the following columns:

**conceptId** integer64: SNOMED CT conceptId (primary key)

**read2\_code** list: character list of 7-character Read 2 codes

**read2\_term** list: character list of Read 2 terms

**ctv3\_concept** list: character list of CTV3 concept codes

**ctv3\_termid** list: character list of CTV3 term description codes

**Value**

A data.table with columns conceptId, read2\_code, ctv3\_concept, ctv3\_termid

**See Also**

READMAPS, getMaps, loadSNOMED

---

loadSNOMED	<i>Load SNOMED CT files from a folder(s) into R data.table objects</i>
------------	--

---

### Description

Identifies relevant SNOMED CT files from the folder structure of a SNOMED CT distribution. This includes the core 'Snapshot' tables mapping tables from the 'Refset' folder and the history substitution table and query table. The relevant tables are loaded into an R environment, which can be saved and then easily retrieved for future use. Files from two folders (e.g. International and UK versions) can be loaded together, and are automatically appended by the function.

### Usage

```
loadSNOMED(folders, active_only = TRUE, version = NULL)
```

### Arguments

folders	Vector of folder paths containing SNOMED CT files
active_only	Whether to limit to current (active) SNOMED CT concepts
version	Version description. If NULL, it is derived from the folder paths and expressed in the form: INT date & UK date

### Details

The SNOMED CT files are available from the NHS Digital Technology Reference Update Distribution: <https://isd.digital.nhs.uk/trud/user/guest/group/0/home>

### Value

An environment containing data.table objects: CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP, REFSET, SIMPLEMAP, EXTENDEDMAP, HISTORY (optional), QUERY (optional)

### See Also

loadREADMAPS, CONCEPT, DESCRIPTION, RELATIONSHIP, STATEDRELATIONSHIP, REFSET, SIMPLEMAP, EXTENDEDMAP, QUERY, HISTORY sampleSNOMED, getSNOMED, exportSNOMEDenvir

### Examples

```
# Create a TEST environment and load the sample dictionaries
TEST <- sampleSNOMED()

# Export to temporary directory
exportSNOMEDenvir(TEST, tempdir())

# Try to import using the loadSNOMED function
```

```
TEST2 <- loadSNOMED(tempdir(), active_only = FALSE)

# Check that reimported SNOMED dictionary is the same as the original
all.equal(TEST$CONCEPT, TEST2$CONCEPT)
all.equal(TEST$DESCRIPTION, TEST2$DESCRIPTION)
all.equal(TEST$RELATIONSHIP, TEST2$RELATIONSHIP)
all.equal(TEST$STATEDRELATIONSHIP, TEST2$STATEDRELATIONSHIP)
all.equal(TEST$REFSET, TEST2$REFSET)
all.equal(TEST$SIMPLEMAP, TEST2$SIMPLEMAP)
all.equal(TEST$EXTENDEDMAP, TEST2$EXTENDEDMAP)
```

---

MANUAL\_SYNONYMS

*Sample manual synonym table to assist in creation of concept database*

---

### Description

Sample manual synonym table to assist in creation of concept database

### Usage

```
data(MANUAL_SYNONYMS)
```

### Format

An object of class "data.table"

### Details

**synonym** character: Synonym, in lower case if case is unimportant

**snomed** character: SNOMED CT term, in lower case if case is unimportant

**bidirectional** boolean: whether synonym can only ever mean snomed. Not to be used for abbreviations or acronyms.

### See Also

createCDB

### Examples

```
data(MANUAL_SYNONYMS)
str(MANUAL_SYNONYMS)
```

---

parents

*Ancestors and descendants of SNOMED CT concepts*

---

### Description

Returns concepts with 'Is a' or inverse 'Is a' relationship with a set of target concepts. Ancestors include parents and all higher relations. Descendants include children and all lower relations.

### Usage

```
parents(  
  conceptIds,  
  include_self = FALSE,  
  SNOMED = getSNOMED(),  
  TRANSITIVE = NULL,  
  ...  
)
```

```
ancestors(  
  conceptIds,  
  include_self = FALSE,  
  SNOMED = getSNOMED(),  
  TRANSITIVE = NULL,  
  ...  
)
```

```
children(  
  conceptIds,  
  include_self = FALSE,  
  SNOMED = getSNOMED(),  
  TRANSITIVE = NULL,  
  ...  
)
```

```
descendants(  
  conceptIds,  
  include_self = FALSE,  
  SNOMED = getSNOMED(),  
  TRANSITIVE = NULL,  
  ...  
)
```

### Arguments

conceptIds	character or integer64 vector of SNOMED concept IDs
include_self	whether to include the original concept(s) in the output, default = FALSE

SNOMED	environment containing a SNOMED dictionary
TRANSITIVE	transitive closure table for ancestors and descendants, containing is-a relationships. This table can be created by <code>createTransitive</code> to speed up the ancestor / descendant functions. If a TRANSITIVE table is provided, the SNOMED environment is not used and <code>relatedConcepts</code> is not called. TRANSITIVE should be a <code>data.table</code> with columns <code>ancestorId</code> and <code>descendantId</code> .
...	other arguments to pass to <code>relatedConcepts</code>

**Value**

a bit64 vector of SNOMED CT concepts

**See Also**

[`createTransitive()`] for creation of TRANSITIVE table, and [`relatedConcepts()`] for the underlying function to extract SNOMED CT relationships.

**Examples**

```
SNOMED <- sampleSNOMED()

parents('Heart failure')
children('Heart failure')
ancestors('Heart failure')
descendants('Heart failure')
```

---

`print.SNOMEDcodelist` *Display a SNOMEDcodelist on screen*

---

**Description**

Displays a SNOMEDcodelist on screen, including metadata. Truncates term descriptions in order to fit within the line width.

**Usage**

```
## S3 method for class 'SNOMEDcodelist'
print(x, ...)
```

**Arguments**

x	SNOMEDcodelist object to print to screen
...	not used

**Value**

invisibly returns the codelist

**See Also**

Other SNOMEDcodelist functions: [SNOMEDcodelist\(\)](#), [addInactiveConcepts\(\)](#), [expandSNOMED\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#)

---

`print.SNOMEDconcept`     *Display a SNOMEDconcept object with descriptions*

---

**Description**

SNOMEDconcept is an S3 class for vectors of SNOMED concept IDs as 64-bit integers. This function checks whether the object has the class SNOMEDconcept and is a vector of 64-bit integers.

**Usage**

```
## S3 method for class 'SNOMEDconcept'  
print(x, ...)
```

**Arguments**

x	SNOMEDconcept object, or something that can be coerced to one
...	not required

**Value**

invisibly returns a character vector of the SNOMED CT concepts with descriptions separated by pipe (|)

**See Also**

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

---

`print.SNOMEDfindings`     *Print method for output of 'decompose' function*

---

**Description**

Print method for output of 'decompose' function

**Usage**

```
## S3 method for class 'SNOMEDfindings'  
print(x, ...)
```

**Arguments**

x SNOMEDfindings object to display  
 ... not used

**Value**

TRUE

---

QUERY

*Sample query table from SNOMED CT dictionary*

---

**Description**

Sample of the SNOMED CT table of ancestor / descendant relationships for inactive concepts.

**Usage**

data(QUERY)

**Format**

An object of class "data.table"

**Details**

**supertypeId** integer64: concept ID of the ancestor (active) concept

**subtypeId** integer64: concept ID of the descendant (inactive) concept

**provenance** integer: provenance of relationship. Provenance = 0 means subsumption is always true. Provenance = 1 means subsumption is usually true (but there is a theoretical risk of false positives). Provenance = 2 means both ancestors and descendents are only approximately known. Provenance = 3 means original code had at least two distinct meanings and all are being returned

**See Also**

Other SNOMEDsample: [HISTORY](#), [SNOMED\\_RELATIONSHIP](#)

**Examples**

```
# Create a TEST environment and load the sample dictionaries
TEST <- new.env()
data(CONCEPT, envir = TEST)
data(DESCRIPTION, envir = TEST)
data(RELATIONSHIP, envir = TEST)
data(STATEDRELATIONSHIP, envir = TEST)
data(QUERY, envir = TEST)

# Show properties of the query table
str(TEST$QUERY)
```



---

READMAPS

*Sample mappings from Read to SNOMED CT*

---

## Description

A sample of a mapping table derived from NHS Digital maps. Contains concepts in Read Clinical Terms Version 2 and Clinical Terms Version 3 that map to a set of SNOMED CT concepts, according to a supplied mapping file. The source data are available from the NHS Digital Technology Reference data Update Distribution <https://isd.digital.nhs.uk/trud/user/guest/group/0/pack/9/subpack/9/releases>.

## Usage

```
data(READMAPS)
```

## Format

An object of class "data.table"

## Details

**conceptId** integer64: SNOMED CT conceptId (primary key)

**read2\_code** list: character list of 7-character Read V2 codes

**read2\_term** list: character list of Read V2 terms

**ctv3\_concept** list: character list of CTV3 concept codes

**ctv3\_termid** list: character list of CTV3 term description codes

## See Also

loadREADMAPS, getMaps

Other sampleSNOMED: [HISTORY](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

## Examples

```
# Show properties of the READMAPS table
data(READMAPS)
str(READMAPS)
```

---

relatedConcepts      *Obtain related concepts for a set of SNOMED CT concepts*

---

### Description

Returns concepts with a particular relation to a supplied set of SNOMED CT concepts

### Usage

```
relatedConcepts(
  conceptIds,
  typeId = bit64::as.integer64("116680003"),
  tables = c("RELATIONSHIP", "STATEDRELATIONSHIP"),
  reverse = FALSE,
  recursive = FALSE,
  active_only = TRUE,
  SNOMED = getSNOMED()
)
```

### Arguments

conceptIds	character or integer64 vector
typeId	concept ID of relationship type. Defaults to 116680003 = Is a
tables	vector of names of relationship table(s) to use; by default use both RELATIONSHIP and STATEDRELATIONSHIP
reverse	whether to reverse the relationship
recursive	whether to re-apply the function on the outputs
active_only	whether to limit the output to active concepts only
SNOMED	environment containing a SNOMED dictionary

### Value

a data.table with the following columns: id, conceptId, type (only if include\_synonyms = TRUE), term, active (only if active\_only = FALSE)

### Examples

```
# Load sample SNOMED CT dictionary
SNOMED <- sampleSNOMED()

# Example: anatomical site of a finding
findingSite <- function(x){
  relatedConcepts(as.SNOMEDconcept(x),
    typeId = as.SNOMEDconcept('Finding site'))
}

description(findingSite('Heart failure'))
# Heart structure (body structure)
```

---

`sampleSNOMED`*Sample SNOMED CT dictionary*

---

**Description**

Returns an environment containing a selection of SNOMED CT terms, their relationships and descriptions which are provided with the package

**Usage**

```
sampleSNOMED()
```

**Value**

environment containing four `data.table` objects: `CONCEPT`, `DESCRIPTION`, `RELATIONSHIP`, `STATEDRELATIONSHIP` and a list named `'metadata'`

**See Also**

`CONCEPT`, `DESCRIPTION`, `RELATIONSHIP`, `STATEDRELATIONSHIP`, `REFSET`, `SIMPLEMAP`, `EXTENDEDMAP`, `HISTORY`, `QUERY`, `loadSNOMED`, `sampleSNOMED`

Other `sampleSNOMED`: [HISTORY](#), [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#)

**Examples**

```
TEST <- sampleSNOMED()
inactiveIncluded(TEST)
SNOMEDconcept('Heart failure', SNOMED = TEST)

# To display metadata for this SNOMED CT dictionary
sampleSNOMED()$metadata
```

---

`semanticType`*Retrieves semantic types using the text 'tag' in the description*

---

**Description**

Uses the fully specified name in the `DESCRIPTION` table. If there are multiple fully specified names, the name with the most recent `effectiveTime` will be used.

**Usage**

```
semanticType(conceptIds, SNOMED = getSNOMED())
```

**Arguments**

conceptIds      character or integer64 vector of SNOMED concept IDs  
 SNOMED          environment containing a SNOMED dictionary

**Value**

a character vector of semantic tags corresponding to the conceptIDs

**Examples**

```
SNOMED <- sampleSNOMED()

semanticType(as.SNOMEDconcept(c('Heart failure', 'Is a')))
```

---

simplify	<i>Retrieves closest single ancestor within a given set of SNOMED CT concepts</i>
----------	---

---

**Description**

Returns a vector of SNOMED CT concept IDs for an ancestor of each concept that is within a second list. If multiple ancestors are included in the second list, the concept is not simplified (i.e. the original concept ID is returned). This functionality can be used to translate concepts into simpler forms for display, e.g. 'Heart failure' instead of 'Heart failure with reduced ejection fraction'.

**Usage**

```
simplify(
  conceptIds,
  ancestorIds,
  SNOMED = getSNOMED(),
  tables = c("RELATIONSHIP", "STATEDRELATIONSHIP")
)
```

**Arguments**

conceptIds      character or integer64 vector of SNOMED concept IDs for concepts for which an ancestor is sought  
 ancestorIds    character or integer64 vector of SNOMED concept IDs for possible ancestors  
 SNOMED          environment containing a SNOMED dictionary  
 tables          character vector of relationship tables to use

**Details**

This function is intended for use with active SNOMED CT concepts only.

**Value**

a data.table with the following columns: originalId (integer64) = original conceptId, ancestorId (integer64) = closest single ancestor, or original concept ID if no ancestor is included among ancestorIds

**Examples**

```
SNOMED <- sampleSNOMED()

original_terms <- c('Systolic heart failure', 'Is a',
  'Heart failure with reduced ejection fraction',
  'Acute kidney injury due to circulatory failure (disorder)')
# Note in this example 'Is a' has no parents in ancestors,
# and acute kidney failure has two parents in ancestors
# so neither of the parents will be chosen.
# Also test out inclusion of duplicate concepts.

ancestors <- simplify(c(as.SNOMEDconcept(original_terms),
  as.SNOMEDconcept(original_terms)[3:4]),
  as.SNOMEDconcept(c('Heart failure', 'Acute heart failure',
  'Cardiorenal syndrome (disorder)')))
print(cbind(original_terms, description(ancestors$ancestorId)$term))
```

---

 SNOMEDcodelist

 Convert a data.frame to a SNOMEDcodelist object
 

---

**Description**

SNOMEDcodelist is an S3 class for lists of SNOMED CT concepts. It consists of conceptId and include\_desc columns. The option to include descendants allows the creation of more succinct SNOMED codelists.

**Usage**

```
SNOMEDcodelist(
  x,
  include_desc = FALSE,
  format = c("simple", "tree", "exptree"),
  codelist_name = NULL,
  version = NULL,
  author = NULL,
  date = NULL,
  SNOMED = getSNOMED(),
  show_excluded_descendants = FALSE
)

as.SNOMEDcodelist(x, ...)
```

**Arguments**

<code>x</code>	vector of SNOMED CT concept IDs, something which can be coerced to a SNOMEDconcept object, or a data.frame with a column 'conceptId' containing SNOMED CT concept IDs in integer64 or text format and optional column 'include_desc' (Boolean) stating whether descendants of the term should be included.
<code>include_desc</code>	Boolean vector stating whether descendants are included, recycled if necessary. Default = FALSE. Ignored if x contains a column 'include_desc'
<code>format</code>	Whether the codelist is expressed as a simple enumeration of concepts ('simple'), as a set of concept hierarchies ('tree'), or concept hierarchies showing all descendant terms ('exptree'). Codelists can be converted between the formats, but the result of conversion may depend on the SNOMED CT dictionary being used.
<code>codelist_name</code>	Name of the codelist (character vector of length 1)
<code>version</code>	Version of the codelist (character vector of length 1)
<code>author</code>	Author of the codelist (character vector of length 1)
<code>date</code>	Date attributed to the codelist (character vector of length 1)
<code>SNOMED</code>	environment containing a SNOMED dictionary
<code>show_excluded_descendants</code>	Whether to show excluded descendants alongside the codes included in the codelist (for a 'tree' or 'exptree' format codelist).
<code>...</code>	other arguments to pass to SNOMEDcodelist

**Details**

Input is a data.frame or data.table with column names 'conceptId' and optionally 'include\_desc', which is FALSE by default, but if TRUE then the codelist automatically includes all active descendants of that concept.

If the codelist is intended to contain inactive concepts, it can only exist in the 'simple' format. Inactive concepts will be lost if the codelist is converted between formats.

as.SNOMEDcodelist converts its argument into a SNOMEDcodelist but leaves it unchanged if it is already a SNOMEDcodelist.

**Value**

An object of class 'SNOMEDcodelist'

**See Also**

htmlCodelistHierarchy

Other SNOMEDcodelist functions: [addInactiveConcepts\(\)](#), [expandSNOMED\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

Other SNOMEDcodelist functions: [addInactiveConcepts\(\)](#), [expandSNOMED\(\)](#), [export\(\)](#), [is.SNOMEDcodelist\(\)](#), [print.SNOMEDcodelist\(\)](#)

**Examples**

```
SNOMED <- sampleSNOMED()

my_concepts <- SNOMEDconcept('Heart failure')
SNOMEDcodelist(my_concepts)
SNOMEDcodelist(data.frame(conceptId = my_concepts))
as.SNOMEDcodelist(data.frame(conceptId = my_concepts,
  include_desc = TRUE))
```

---

SNOMED_CONCEPT	<i>Sample concept table from SNOMED CT dictionary</i>
----------------	---

---

**Description**

A sample of the SNOMED CT concept table.

**Usage**

```
data(CONCEPT)
```

**Format**

An object of class "data.table"

**Details**

**id** integer64: SNOMED CT conceptId (primary key)

**moduleId** integer64: class of SNOMED CT concept (whether it is used for recording information or is a metadata concept)

**definitionStatusId** integer64: 900000000000074008 = primitive concept, 900000000000073002 = defined by conditions

**effectiveTime** IDate: when the concept became active

**active** logical: whether this concept is currently active

**See Also**

Other sampleSNOMED: [HISTORY](#), [READMAPS](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

**Examples**

```
# Show properties of the CONCEPT table
data('CONCEPT')
str(CONCEPT)
```

---

SNOMED\_DESCRIPTION      *Sample description table from SNOMED CT dictionary*

---

### Description

A sample of the SNOMED CT description table. Each concept may has a fully specified name and may have any number of synonyms.

### Usage

```
data(DESCRIPTION)
```

### Format

An object of class "data.table"

### Details

**id** integer64: description ID

**moduleId** integer64: class of SNOMED CT concept (whether it is used for recording information or is a metadata concept)

**conceptId** integer64: SNOMED CT concept ID

**languageCode** character: 'en' = English

**typeId** integer64: 900000000000013009 = Synonym, 90000000000003001 = Fully Specified Name

**term** character: term description

**caseSignificanceId** integer64: 90000000000020002 = Initial character case sensitive, 90000000000017005 = Whole term case sensitive, 900000000000448009 = Whole term case insensitive

**effectiveTime** IDate: when the concept became active

**active** logical: whether this concept is currently active

### See Also

Other sampleSNOMED: [HISTORY](#), [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

### Examples

```
# Show properties of the DESCRIPTION table
data('DESCRIPTION')
str(DESCRIPTION)
```



---

SNOMED\_EXTENDEDMAP     *Sample extended map table from SNOMED CT dictionary*

---

### Description

A sample of the SNOMED CT extended map table, containing maps to ICD-10 and OPCS4.

### Usage

```
data(EXTENDEDMAP)
```

### Format

An object of class "data.table"

### Details

**moduleId** integer64: core metadata concept: 449080006 = SNOMED CT to ICD-10 rule-based mapping module, 999000031000000106 = SNOMED CT United Kingdom Edition reference set module

**refsetId** integer64: foundation metadata concept: 447562003 = ICD-10 complex map reference set, 1126441000000105 = Office of Population Censuses and Surveys Classification of Interventions and Procedures Version 4.9 complex map reference set, 999002271000000101 = International Classification of Diseases, Tenth Revision, Fifth Edition, five character code United Kingdom complex map reference set

**referencedComponentId** integer64: SNOMED CT conceptId of the concept mapped

**mapGroup** integer: mapping group

**mapPriority** integer: priority of alternative maps (1 = highest)

**mapRule** character: advice on mapping rule

**mapAdvice** character: mapping advice

**mapTarget** character: target ICD-10 or OPCS4 code. The optional period between the third and fourth character has been removed for consistency.

**mapCategoryId** integer64: foundation metadata concept describing the quality of the map

**effectiveTime** IDate: when the concept became active

**active** logical: whether this concept is currently active

### See Also

Other sampleSNOMED: [HISTORY](#), [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

**Examples**

```
# Load the dataset and show its properties
data('EXTENDEDMAP')
str(EXTENDEDMAP)

# This EXTENDEDMAP table is part of the sample SNOMED CT dictionary
# Hence this should show the same properties as above
str(sampleSNOMED())$EXTENDEDMAP
```

---

SNOMED_REFSET	<i>Sample refset table from SNOMED CT dictionary</i>
---------------	--

---

**Description**

A sample of the SNOMED CT refset table. This contains SNOMED CT codelists that are used for particular operational or clinical purposes, and are curated by SNOMED CT. The id column of the refset table is not included, in order to save space.

**Usage**

```
data(REFSET)
```

**Format**

An object of class "data.table"

**Details**

**moduleId** integer64: SNOMED CT core metadata concept, stating whether the refset is from the SNOMED CT core module or the UK extension.

**refsetId** integer64: SNOMED CT conceptId of the refset. These concepts have semantic type 'foundation metadata concept'

**referencedComponentId** integer64: SNOMED CT conceptId of the member of the refset

**effectiveTime** IDate: when the concept became active

**active** logical: whether this concept is currently active

**See Also**

Other sampleSNOMED: [HISTORY](#), [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_RELATIONSHIP](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

**Examples**

```
# Load the dataset and show its properties
data('REFSET')
str(REFSET)

# This REFSET table is part of the sample SNOMED CT dictionary
# Hence this should show the same properties as above
str(sampleSNOMED())$REFSET)
```

---

SNOMED\_RELATIONSHIP     *Sample relationship tables from SNOMED CT dictionary*

---

**Description**

Samples of the SNOMED CT tables of stated relationships (RELATIONSHIP) and inferred relationships (RELATIONSHIP).

**Usage**

```
data(RELATIONSHIP); data(STATEDRELATIONSHIP)

STATEDRELATIONSHIP
```

**Format**

An object of class "data.table"

An object of class data.table (inherits from data.frame) with 329 rows and 10 columns.

**Details**

**id** integer64: ID of the relationship record (primary key)

**active** logical: whether this concept is currently active

**moduleId** integer64: class of SNOMED CT concept (whether it is used for recording information or is a metadata concept)

**sourceId** integer64: source SNOMED CT concept for the relationship

**destinationId** integer64: destination SNOMED CT concept for the relationship

**relationshipGroup** integer: group ID for relationships that are grouped

**characteristicTypeId** integer64: 900000000000011006 = Inferred relationship

**modifierId** integer64: 9000000000000451002 = Existential restriction modifier

**effectiveTime** IDate: when the concept became active

**typeId** integer64: type of relationship, e.g. 116680003 = Is a, 42752001 = Due to, 246090004 = Associated finding, 363698007 = Finding site, 363702006 = Has focus

**See Also**

Other SNOMEDsample: [HISTORY](#), [QUERY](#)

Other sampleSNOMED: [HISTORY](#), [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_SIMPLEMAP](#), [sampleSNOMED\(\)](#)

**Examples**

```
# Create a TEST environment and load the sample dictionaries
TEST <- new.env()
data(CONCEPT, envir = TEST)
data(DESCRIPTION, envir = TEST)
data(RELATIONSHIP, envir = TEST)
data(STATEDRELATIONSHIP, envir = TEST)

# Show properties of the relationship tables
str(TEST$RELATIONSHIP)
str(TEST$STATEDRELATIONSHIP)
```

---

SNOMED_SIMPLEMAP	<i>Sample SIMPLE map table from SNOMED CT dictionary</i>
------------------	--

---

**Description**

A sample of the SNOMED CT SIMPLE map table, containing maps to ICD-10 and OPCS4.

**Usage**

```
data(SIMPLEMAP)
```

**Format**

An object of class "data.table"

**Details**

**moduleId** integer64: core metadata concept: 90000000000207008 = SNOMED CT core module, 999000021000000109 = SNOMED CT United Kingdom clinical extension reference set module, 999000031000000106 = SNOMED CT United Kingdom Edition reference set module

**refsetId** integer64: foundation metadata concept: 900000000000497000 = CTV3 simple map reference set, 446608001 = ICD-O simple map reference set, 1323081000000108 = Coronavirus disease 19 caused by severe acute respiratory syndrome coronavirus 2 test result communication to general practice concept simple map reference set, 1323091000000105 = Coronavirus disease 19 caused by severe acute respiratory syndrome coronavirus 2 test result communication to general practice description simple map reference set, 82551000000107 = National Health Service England National Genomic Test Directory whole genome sequencing test simple map reference set

**referencedComponentId** integer64: SNOMED CT conceptId of the concept mapped

**mapTarget** character: target ICD-O or CTV3 code  
**effectiveTime** IDate: when the concept became active  
**active** logical: whether this concept is currently active

### See Also

Other sampleSNOMED: [HISTORY](#), [READMAPS](#), [SNOMED\\_CONCEPT](#), [SNOMED\\_DESCRIPTION](#), [SNOMED\\_EXTENDEDMAP](#), [SNOMED\\_REFSET](#), [SNOMED\\_RELATIONSHIP](#), [sampleSNOMED\(\)](#)

### Examples

```
# Load the dataset and show its properties
data('SIMPLEMAP')
str(SIMPLEMAP)

# This SIMPLEMAP table is part of the sample SNOMED CT dictionary
# Hence this should show the same properties as above
str(sampleSNOMED()$SIMPLEMAP)
```

---

union.SNOMEDconcept    *Set operations for SNOMEDconcept vectors*

---

### Description

The default set functions in the base package do not handle integer64 vectors correctly, so this package also provides new generic functions for union, intersect and setdiff, which enable the appropriate object-specific function to be called according to the class of the vector. This means that SNOMEDconcept vectors will remain as SNOMEDconcept vectors when these functions are used.

### Usage

```
## S3 method for class 'SNOMEDconcept'
union(x, y)

union(x, y)

## Default S3 method:
union(x, y)

## S3 method for class 'SNOMEDconcept'
intersect(x, y)

intersect(x, y)

## Default S3 method:
intersect(x, y)
```

```
## S3 method for class 'SNOMEDconcept'
setdiff(x, y)

setdiff(x, y)

## Default S3 method:
setdiff(x, y)
```

### Arguments

x	SNOMEDconcept vector
y	SNOMEDconcept vector, or an object that can be coerced to SNOMEDconcept by as.SNOMEDconcept

### Value

an integer64 vector of SNOMEDconcept class

### See Also

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [unique.SNOMEDconcept\(\)](#)

### Examples

```
sys_acute <- SNOMEDconcept(c('Systolic heart failure',
  'Acute heart failure'), SNOMED = sampleSNOMED())
acute_left_right <- SNOMEDconcept(c('Acute heart failure',
  'Left heart failure', 'Right heart failure'),
  SNOMED = sampleSNOMED())
union(sys_acute, acute_left_right)
intersect(sys_acute, acute_left_right)
setdiff(sys_acute, acute_left_right)
```

---

unique.SNOMEDconcept *Unique vector of SNOMED CT concepts*

---

### Description

SNOMEDconcept is an S3 class for vectors of SNOMED concept IDs as 64-bit integers. This function returns a vector containing only unique SNOMEDconcept values.

### Usage

```
## S3 method for class 'SNOMEDconcept'  
unique(x, ...)
```

### Arguments

x	SNOMEDconcept vector
...	other variables to pass on to the underlying 'unique' function

### Value

SNOMEDconcept vector with duplicates removed

### See Also

Other SNOMEDconcept functions: [as.data.frame.SNOMEDconcept\(\)](#), [c.SNOMEDconcept\(\)](#), [is.SNOMEDconcept\(\)](#), [print.SNOMEDconcept\(\)](#), [union.SNOMEDconcept\(\)](#)

### Examples

```
hf <- SNOMEDconcept('Heart failure', SNOMED = sampleSNOMED())  
hf2 <- c(hf, hf)  
unique(hf2)
```

# Index

- \* **CDB functions**
  - addComposeLookupToCDB, 4
  - addWordnet, 5
  - batchDecompose, 9
  - createCDB, 12
  - downloadOrphanet, 17
  - downloadWordnet, 18
- \* **MiADE functions**
  - exclude\_irrelevant\_findings, 19
  - exportMiADECDB, 22
- \* **SNOMEDcodelist functions**
  - addInactiveConcepts, 5
  - expandSNOMED, 20
  - export, 21
  - is.SNOMEDcodelist, 32
  - print.SNOMEDcodelist, 38
  - SNOMEDcodelist, 45
- \* **SNOMEDconcept functions**
  - as.data.frame.SNOMEDconcept, 7
  - c.SNOMEDconcept, 10
  - is.SNOMEDconcept, 33
  - print.SNOMEDconcept, 39
  - union.SNOMEDconcept, 53
  - unique.SNOMEDconcept, 55
- \* **SNOMEDsample**
  - HISTORY, 29
  - QUERY, 40
  - SNOMED\_RELATIONSHIP, 51
- \* **datasets**
  - HISTORY, 29
  - MANUAL\_SYNONYMS, 36
  - QUERY, 40
  - READMAPS, 41
  - SNOMED\_CONCEPT, 47
  - SNOMED\_DESCRIPTION, 48
  - SNOMED\_EXTENDEDMAP, 49
  - SNOMED\_REFSET, 50
  - SNOMED\_RELATIONSHIP, 51
  - SNOMED\_SIMPLEMAP, 52
- \* **sampleSNOMED**
  - HISTORY, 29
  - READMAPS, 41
  - sampleSNOMED, 43
  - SNOMED\_CONCEPT, 47
  - SNOMED\_DESCRIPTION, 48
  - SNOMED\_EXTENDEDMAP, 49
  - SNOMED\_REFSET, 50
  - SNOMED\_RELATIONSHIP, 51
  - SNOMED\_SIMPLEMAP, 52
- acronyms, 3
- addComposeLookupToCDB, 4, 6, 9, 13, 17, 18
- addInactiveConcepts, 5, 20, 21, 32, 39, 46
- addWordnet, 4, 5, 9, 13, 17, 18
- ancestors (parents), 37
- as.data.frame.SNOMEDconcept, 7, 10, 33, 39, 54, 55
- as.integer64.SNOMEDconcept  
(as.data.frame.SNOMEDconcept), 7
- as.SNOMEDcodelist (SNOMEDcodelist), 45
- as.SNOMEDconcept  
(as.data.frame.SNOMEDconcept), 7
- attrConcept, 8
- batchDecompose, 4, 6, 9, 13, 17, 18
- blacklist\_almost\_all\_except\_diseases  
(exclude\_irrelevant\_findings), 19
- blacklist\_vague\_findings  
(exclude\_irrelevant\_findings), 19
- c.SNOMEDconcept, 8, 10, 33, 39, 54, 55
- children (parents), 37
- compose, 10
- CONCEPT (SNOMED\_CONCEPT), 47
- contractSNOMED (expandSNOMED), 20



- createCDB, [4](#), [6](#), [9](#), [12](#), [17](#), [18](#)
- createSNOMEDindices, [13](#)
- createTransitive, [14](#)
- decompose, [14](#)
- descendants (parents), [37](#)
- DESCRIPTION (SNOMED\_DESCRIPTION), [48](#)
- description, [16](#)
- downloadOrphanet, [4](#), [6](#), [9](#), [13](#), [17](#), [18](#)
- downloadWordnet, [4](#), [6](#), [9](#), [13](#), [17](#), [18](#)
- exclude\_irrelevant\_findings, [19](#), [23](#)
- expandSNOMED, [5](#), [20](#), [21](#), [32](#), [39](#), [46](#)
- export, [5](#), [20](#), [21](#), [21](#), [32](#), [39](#), [46](#)
- exportMiADECDB, [19](#), [22](#)
- exportSNOMEDenvir, [24](#)
- EXTENDEDMAP (SNOMED\_EXTENDEDMAP), [49](#)
- getMaps, [24](#)
- getRefset, [26](#)
- getSNOMED, [27](#)
- hasAttributes, [28](#)
- HISTORY, [29](#), [40](#), [41](#), [43](#), [47–50](#), [52](#), [53](#)
- htmlCodelistHierarchy, [30](#)
- inactiveIncluded, [31](#)
- intersect (union.SNOMEDconcept), [53](#)
- is.SNOMEDcodelist, [5](#), [20](#), [21](#), [32](#), [39](#), [46](#)
- is.SNOMEDconcept, [8](#), [10](#), [33](#), [39](#), [54](#), [55](#)
- loadREADMAPS, [33](#)
- loadSNOMED, [35](#)
- MANUAL\_SYNONYMS, [36](#)
- parents, [37](#)
- print.SNOMEDcodelist, [5](#), [20](#), [21](#), [32](#), [38](#), [46](#)
- print.SNOMEDconcept, [8](#), [10](#), [33](#), [39](#), [54](#), [55](#)
- print.SNOMEDfindings, [39](#)
- QUERY, [29](#), [40](#), [52](#)
- READMAPS, [29](#), [41](#), [43](#), [47–50](#), [52](#), [53](#)
- REFSET (SNOMED\_REFSET), [50](#)
- relatedConcepts, [42](#)
- RELATIONSHIP (SNOMED\_RELATIONSHIP), [51](#)
- sampleSNOMED, [29](#), [41](#), [43](#), [47–50](#), [52](#), [53](#)
- semanticType, [43](#)
- setdiff (union.SNOMEDconcept), [53](#)
- showCodelistHierarchy (expandSNOMED), [20](#)
- SIMPLEMAP (SNOMED\_SIMPLEMAP), [52](#)
- simplify, [44](#)
- SNOMED\_CONCEPT, [29](#), [41](#), [43](#), [47](#), [48–50](#), [52](#), [53](#)
- SNOMED\_DESCRIPTION, [29](#), [41](#), [43](#), [47](#), [48](#), [49](#), [50](#), [52](#), [53](#)
- SNOMED\_EXTENDEDMAP, [29](#), [41](#), [43](#), [47](#), [48](#), [49](#), [50](#), [52](#), [53](#)
- SNOMED\_REFSET, [29](#), [41](#), [43](#), [47–49](#), [50](#), [52](#), [53](#)
- SNOMED\_RELATIONSHIP, [29](#), [40](#), [41](#), [43](#), [47–50](#), [51](#), [53](#)
- SNOMED\_SIMPLEMAP, [29](#), [41](#), [43](#), [47–50](#), [52](#), [52](#)
- SNOMED\_STATEDRELATIONSHIP (SNOMED\_RELATIONSHIP), [51](#)
- SNOMEDcodelist, [5](#), [20](#), [21](#), [32](#), [39](#), [45](#)
- SNOMEDconcept (as.data.frame.SNOMEDconcept), [7](#)
- STATEDRELATIONSHIP (SNOMED\_RELATIONSHIP), [51](#)
- union (union.SNOMEDconcept), [53](#)
- union.SNOMEDconcept, [8](#), [10](#), [33](#), [39](#), [53](#), [55](#)
- unique.SNOMEDconcept, [8](#), [10](#), [33](#), [39](#), [54](#), [55](#)