Using CemConnector

2021-07-28

Contents

Introduction

If you're reading this guide it is expected that you have some familiarity with the common evidence model (CEM) and have an installation set up and configured. Please refer to (the CEM github repository)[https://github.com/OHDSI/CommonEvidenceModel] for more information.

The following guide highlights how to use the CemConnector package (following installation instructions provided in the readme).

Connecting

CemConnector provides two methods to connect to a Cem that are functionally equivalent: connecting via a web api and connecting via a database.

Using CEMConnector with a web backend

The simplest way to connect with the CEM is to connect to a host web instance, for example the public API available at https://cem.ohdsi.org/ (NOTE: TBD). To do this we will use a web backend. First we need to create a connection to the web api:

```
cemConnection <- CemConnector::CemWebApiBackend(apiUrl = "https://cem.ohdsi.org/")
```

Using CEMConnector with a database backend

Alternatively, a connection can be made with a DatabaseConnector::connectionDetails object. Contact your CEM administrator for details connecting this way.

```
connectionDetails <- DatabaseConnector::createConnectionDetails(user = "mydbusername",
                                                                server = "myserver/foo",
                                                                dbms = "redshift",
                                                                password = "mysecret")

cemConnection <- CemConnector::CemDatabaseBackend$new(connectionDetails = connectionDetails,
                                                      vocabularySchema = "vocabulary",
                                                      cemSchema = "cem_v2",
                                                      sourceSchema = "cem_v2_source")
```

This should connect to the server without error.

Viewing evidence sources

The available evidence sources in the Cem are visible as follows:

```
cemConnection$getCemSourceInfo()
#> # A tibble: 10 x 9
#>    sourceId  description   provenance contributorOrgan~ contactName creationDate
#>    <chr>     <chr>         <chr>      <chr>              <chr>       <date>
#>  1 AEOLUS    "Spontaneous~ AEOLUS     Center for Biome~ Lee Evans ~ 2021-01-02
#>  2 COMMONEV~ "CommonEvide~ COMMONEVI~ OHDSI              Erica Voss  2020-02-26
#>  3 EU_PL_ADR "From the PR~ EU_PL_ADR  PROTECT            PROTECT     2020-12-20
#>  4 MEDLINE_~ "Co-occurren~ MEDLINE    Janssen R&D        Erica Voss  2020-02-09
#>  5 MEDLINE_~ "Co-occurren~ PUBMED     Janssen R&D        Erica Voss  2020-02-19
#>  6 MEDLINE_~ "Winnenburg ~ MEDLINE    Janssen R&D        Erica Voss  2020-02-19
#>  7 OMOP_VOC~ "OMOP Vocabu~ VOCABULARY OHDSI              Odysseus    2020-08-12
#>  8 SEMMEDDB  "Semantic Me~ SEMMEDDB   National Institu~ National I~ 2020-12-20
#>  9 SHERLOCK  "ClinicalTri~ SHERLOCK   Janssen R&D        Erica Voss  2020-01-27
#> 10 SPLICER   "Adverse dru~ SPLICER    Georgia Tech       Jon Duke    2019-12-08
#> # ... with 3 more variables: coverageStartDate <date>, coverageEndDate <date>,
#> #   versionIdentifier <chr>
```

This highlights who, what, where and when the data sources come from.

Constructing conceptset defintions

To generate concept sets we recommend using PHEOBE. This is a simple way of finding standard concepts (SNOMED or RxNorm) that relate to diseases and outcomes of interest for which real world evidence has been observed. For this example we will use the concept for moderate depression as a disease outcome and the ingredient exposure codeine. It is important to note that the CEM stores information in the standard vocabulary only. Similarly drug concepts are mapped at the ingredient level, so you will want to map specific ingredient concepts not drug formulations. For example, instead of using a specific drug dose such as 'codeine 30 MG' it would be advisable to use only the ingredient 'codeine'. Naturally, designing concept sets is highly specific to a phenotype of interest and careful consideration should go into the concepts used.

In this example, we would like to both find any existing evidence that relates these two diseases as well as select potential negative controls that can be used to calibrate p-values and confidence intervals from effect estimates that we may generate in observational studies.

Concept sets can also be imported from ATLAS. See the ROhdsiWebApi package for more information on how to use this in R.

CemConnector requires 3 fields for searching for relationships: `conceptId`, `includeDescendants` and `isExcluded`. Other fields can be present in the `data.frame`, but will be unused in searches.

Negative control suggestions

Negative controls are one of the most common ways to use the information within the CEM in observational health research. Cem Connector is capable of suggesting negative controls for use in population level effect estimate studies. Negative controls are intended to be exposures or outcomes selected because they are believed to have no direct causal relationship with the outcome of interest. For example, in-grown toe-nails are not caused by exposure to ACE inhibitors. However, in-grown toenails are extremely common, and many patients will experience this outcome while exposed to the drug. Consequently, any observed effect is a product of residual bias (e.g. from the study design choices) and this can be used to produce calibrated effect estimates.

The design of a PLE study is outside the scope of this viggnette, however, the selection of outcomes/exposures can be achieved at the concept level. Here, the CemConnector takes a concept set as a parameter and finds outcomes our exposures that have no mapped relationships within the CEM. We note that these mappings are not perfect and human curation by experienced clinicians is advised.

To find negative control outcomes for the above ingredient set use the following code, for example the drug `Codene`.

```
ingredientConceptSet <- data.frame(conceptId = c(1201620), includeDescendants = c(1), isExcluded = c(0))
outcomeConcepts <- cemConnection$getSuggestedControlCondtions(ingredientConceptSet)
outcomeConcepts
#> # A tibble: 50 x 2
#>    conditionConceptId conceptName
#>                 <int> <chr>
#>  1             436070 Vitamin D deficiency
#>  2              80816 Degeneration of intervertebral disc
#>  3             374375 Impacted cerumen
#>  4            4079750 Osteoarthritis of knee
#>  5             261880 Atelectasis
#>  6             201826 Type 2 diabetes mellitus
#>  7             139099 Ingrowing nail
#>  8             381877 Dysfunction of eustachian tube
#>  9             379805 Myopia
#> 10             377910 Deviated nasal septum
#> # ... with 40 more rows
```

Alternatively, the use of ingredient exposures may be considered (i.e. drugs thought not to cause or treat a given condition). For example, when searching for depressive disorder and descendants:

```
conditionConceptSet <- data.frame(conceptId = c(440383), includeDescendants = c(1), isExcluded = c(0))
outcomeConcepts <- cemConnection$getSuggestedControlIngredients(conditionConceptSet)
outcomeConcepts
#> # A tibble: 0 x 2
#> # ... with 2 variables: ingredientConceptId <int>, conceptName <chr>
```

In both cases the row order indicates the rank of the negative control suggestions. This is simply how commonly a co-occurence has been observed in the OHDSI network. For example, `in-grown toe nails` and `acetominophen` are very commonly observed, yet will likely have no medical relationship between drugs and patient outcomes, respectively.

Exploring evidence

The cem contains a It is often desirable to find out what information relates exposures and outcomes. The cem provides a variety of information, including drug product labels, literature searches, clinical trials data and spontaneous adverse event reporting data.

Searching for conditions related to exposures

The simplest drug concept set is of one ingredient, for example the drug `Codene`.

```
ingredientConceptSet <- data.frame(conceptId = c(1201620), includeDescendants = c(1), isExcluded = c(0))
outcomeConcepts <- cemConnection$getIngredientEvidenceSummary(ingredientConceptSet)
```

Though not necessary for this ingredient set, we allow the utility to look for descendants. This may be useful for an entire class of medications such as the ATC classification `Other opioids`:

```
parentIngredientSet <- data.frame(conceptId = c(21604296), includeDescendants = c(1), isExcluded = c(0))
```

We would like to search for any related disease concepts to this query. Cem connector provides a simple evidence summary for all resulting concepts that are in the database.

```
outcomeConcepts <- cemConnection$getIngredientEvidenceSummary(parentIngredientSet)
outcomeConcepts
#> # A tibble: 5,092 x 3
#>    conditionConceptId conceptName                evidenceExists
#>                 <int> <chr>                               <int>
#>  1              22350 Edema of larynx                         1
#>  2              22426 Congenital macrostomia                  0
#>  3              22820 Tuberculosis of esophagus               1
#>  4              22955 Perforation of esophagus                1
#>  5              23034 Neonatal hypoglycemia                   1
#>  6              23137 Chlamydial pharyngitis                  1
#>  7              23220 Chronic tonsillitis                     1
#>  8              23245 Esophageal bleeding                     1
#>  9              23304 Neck webbing                            1
#> 10              23325 Heartburn                               1
#> # ... with 5,082 more rows
```

The resulting data frame is of ingredient concepts with a 1 or 0 if any evidence exists within the CEM.

For an item here, we may want to look at the specific evidence that is involved. To do this we would look at an exposure outcome pair from the concept set.

To get all the evidence found in the CEM for the ingredient concept set, what source its from and what the counts or statistics are, we can look at the entire search universe.

```
conditionEvidence <- cemConnection$getIngredientEvidence(parentIngredientSet)
conditionEvidence
#> # A tibble: 41 x 16
#>    conceptName1 conceptName2   conceptId1 sourceCode1 sourceCodeType1 conceptId2
#>    <chr>        <chr>               <int> <chr>       <chr>               <int>
#>  1 Codeine      Jaundice          1201620 1201620     OMOP CONCEPT_ID     137977
#>  2 Codeine      Jaundice          1201620 1201620     OMOP CONCEPT_ID     137977
#>  3 Codeine      Jaundice          1201620 1201620     OMOP CONCEPT_ID     137977
#>  4 Codeine      Jaundice          1201620 1201620     OMOP CONCEPT_ID     137977
#>  5 Codeine      Dysthymia         1201620 1201620     OMOP CONCEPT_ID     433440
#>  6 Codeine      Dysthymia         1201620 1201620     OMOP CONCEPT_ID     433440
#>  7 Codeine      Dysthymia         1201620 1201620     OMOP CONCEPT_ID     433440
#>  8 Codeine      Dysthymia         1201620 1201620     OMOP CONCEPT_ID     433440
#>  9 Codeine      Feeling nervo~    1201620 1201620     OMOP CONCEPT_ID     436817
#> 10 Codeine      Feeling nervo~    1201620 1201620     OMOP CONCEPT_ID     436817
#> # ... with 31 more rows, and 10 more variables: sourceCode2 <chr>,
#> #   sourceCodeType2 <chr>, sourceId <chr>, evidenceType <chr>,
#> #   relationshipId <chr>, statisticValue <dbl>, statisticValueType <chr>,
#> #   uniqueIdentifier <chr>, uniqueIdentifierType <chr>, countHow <chr>
```

Searching for drug ingredients related to conditions

Searching for conditions can be achieved similar manner. In this example we search for evidence related to Dysthymia.

```
conditionConceptSet <- data.frame(conceptId = c(433440), includeDescendants = c(1), isExcluded = c(0))
cemConnection$getConditionEvidenceSummary(ingredientConceptSet)
#> # A tibble: 0 x 3
#> # ... with 3 variables: ingredientConceptId <int>, conceptName <chr>,
#> #   evidenceExists <lgl>
```

```
cemConnection$getConditionEvidence(ingredientConceptSet)
#> # A tibble: 0 x 16
#> # ... with 16 variables: conceptName1 <chr>, conceptName2 <chr>,
#> #   conceptId1 <int>, sourceCode1 <chr>, sourceCodeType1 <chr>,
#> #   conceptId2 <int>, sourceCode2 <chr>, sourceCodeType2 <chr>, sourceId <chr>,
#> #   evidenceType <chr>, relationshipId <chr>, statisticValue <dbl>,
#> #   statisticValueType <chr>, uniqueIdentifier <chr>,
#> #   uniqueIdentifierType <chr>, countHow <chr>
```

The mapping between conditions and ingredients is often complicated, for example moderate depressive disorder may not match exactly. Consequently, it may be desirable to explore `siblings` of the related concept, using the concept ancestory. This can be achieved with the `siblingLookupLevels` parameter, which is an integer tha can be used to search higher levels in the concept ancestry:

```
minorDepressionConceptSet <- data.frame(conceptId = c(440383), includeDescendants = c(0), isExcluded = c
cemConnection$getConditionEvidence(minorDepressionConceptSet, siblingLookupLevels = 1)
#> # A tibble: 0 x 16
#> # ... with 16 variables: conceptName1 <chr>, conceptName2 <chr>,
#> #   conceptId1 <int>, sourceCode1 <chr>, sourceCodeType1 <chr>,
#> #   conceptId2 <int>, sourceCode2 <chr>, sourceCodeType2 <chr>, sourceId <chr>,
#> #   evidenceType <chr>, relationshipId <chr>, statisticValue <dbl>,
```

```
#> #   statisticValueType <chr>, uniqueIdentifier <chr>,
#> #   uniqueIdentifierType <chr>, countHow <chr>
```

However, this approach is likely to result in low specificity, when searching for related concepts. It is strongly advised to consider a revised concept set, especially when searching for negative controls.

Searching for evidence pairs

The relationships between specific ingredient and condition based concept sets are often highly informative. A goal of Cem connector is allowed exploration of this evidence. For example, we may be interested in what evidence (if any) relates `codene` with