Creating covariates based on other cohorts

Martijn J. Schuemie

2024-07-15

Contents

1 Introduction

This vignette assumes you are already familiar with the `FeatureExtraction` package.

The `FeatureExtraction` package can generate a default set of covariates, such as one covariate for each condition found in the `condition_occurrence` table. These covariates are based on the concepts found in the data in the Common Data Model (CDM). For example, if we wish to have a covariate for 'prior diabetes', the default set of covariates includes a covariate based on the condition concept `Diabetes melitus` and all of its descendant concepts.

In this vignette we review how we can create covariates based on other cohorts instead of concepts by themselves. In our diabetes example, we may wish to define diabetes not just as the occurrence of a diagnose code, but also require records of anti-diabetic treatments, or blood glucose values exceeding certain thresholds. We can construct a complicated cohort definition, for example using the OHDSI ATLAS tool, encoding this logic. We can instantiate this cohort, and then construct covariates based on the presence or absence of this cohort in a patient's history.

2 Overview

To construct covariates based on other cohorts, the following steps must be taken:

1. Populate a table with the cohorts to be used for covariate construction.
2. Use the `createCohortBasedCovariateSettings()` or `createCohortBasedTemporalCovariateSettings()` function to create a `covariateSettings` object pointing to the cohorts mentioned in the previous steps.

2.1 Populate a table with the cohorts to be used for covariate construction.

The cohorts should be loaded into a cohort table with the standard fields. **This can be the same table as the one containing the main cohorts**. The table should at least have the following fields:

- `cohort_definition_id`, A unique identifier for the cohort. This ID will be used (together with the analysis ID) to construct the covariate ID, so **ensure that the `cohort_definition_id` fits in a 32-bit integer*** (so between -2,147,483,648 and 2,147,483,647).
- `subject_id`, The unique identifier for the person. Should match the `person_id` in the CDM.
- `cohort_start_date`, The date the person enters the cohort.
- `cohort_end_date`, The date the the person exits the cohort. If `null`, the person is assumed to exit the cohort on the same date as entering it.

A person can enter and exit a cohort multiple times, but cannot be in the same cohort at the same time multiple times (this is true for all cohorts in OHDSI).

3 Example

3.1 Creating the cohort attributes and attributes definitions

In this example we will create two cohorts: the main cohort is people initiating diclofenac treatment for the first time. The cohort we wish to use to construct a covariate is type 2 diabetes, requiring both a diagnosis code and a treatment code (in the 30 days following the diagnosis). The diabetes cohort is assumed to start at the first diagnosis meeting the criteria, and end when observation ends (chronic).

```
/************************
File covariateCohorts.sql
************************/
DROP TABLE IF EXISTS @cohort_database_schema.@cohort_table;

CREATE TABLE @cohort_database_schema.@cohort_table (
    cohort_definition_id INT,
    subject_id BIGINT,
    cohort_start_date DATE,
    cohort_end_date DATE
    );

INSERT INTO @cohort_database_schema.@cohort_table (
    cohort_definition_id,
    subject_id,
    cohort_start_date,
    cohort_end_date
    )
SELECT 1,
    person_id,
    MIN(drug_era_start_date),
    MIN(drug_era_end_date)
FROM @cdm_database_schema.drug_era
WHERE drug_concept_id = 1124300 --diclofenac
GROUP BY person_id;

INSERT INTO @cohort_database_schema.@cohort_table (
```

```
    cohort_definition_id,
    subject_id,
    cohort_start_date,
    cohort_end_date
    )
SELECT 2,
    condition_occurrence.person_id,
    MIN(condition_start_date),
    MIN(observation_period_end_date)
FROM @cdm_database_schema.condition_occurrence
INNER JOIN @cdm_database_schema.drug_exposure
    ON condition_occurrence.person_id = drug_exposure.person_id
        AND drug_exposure_start_date >= condition_start_date
        AND drug_exposure_start_date < DATEADD(DAY, 30, condition_start_date)
INNER JOIN @cdm_database_schema.observation_period
    ON condition_occurrence.person_id = observation_period.person_id
        AND condition_start_date >= observation_period_start_date
        AND condition_start_date <= observation_period_end_date
WHERE condition_concept_id IN (
        SELECT descendant_concept_id
        FROM @cdm_database_schema.concept_ancestor
        WHERE ancestor_concept_id = 201826 -- Type 2 diabetes mellitus
        )
    AND drug_concept_id IN (
        SELECT descendant_concept_id
        FROM @cdm_database_schema.concept_ancestor
        WHERE ancestor_concept_id = 21600712 -- DRUGS USED IN DIABETES (ATC A10)
        )
GROUP BY condition_occurrence.person_id;
```

We substitute the arguments in this SQL with actual values, translate it to the right SQL dialect, and execute the SQL:

```
library(SqlRender)
sql <- readSql("covariateCohorts.sql")
connection <- connect(connectionDetails)
renderTranslateExecuteSql(
  connection = connection,
  sql = sql,
  cdm_database_schema = cdmDatabaseSchema,
  cohort_database_schema = cohortDatabaseSchema,
  cohort_table = cohortTable
)
```

If all went well, we now have a table with the cohorts. We can see how many cohorts per type:

```
sql <- paste(
  "SELECT cohort_definition_id,
                COUNT(*) AS count",
  "FROM @cohort_database_schema.@cohort_table",
  "GROUP BY cohort_definition_id"
)
renderTranslateQuerySql(
```

```
  connection = connection,
  sql = sql,
  cohort_database_schema = cohortDatabaseSchema,
  cohort_table = cohortTable
)
```

```
##   cohort_concept_id  count
## 1                 1 954179
## 2                 2 979874
```

3.2 Using the cohort as covariate

To use the constructed diabetes cohort as a covariate, we need to create a `covariateSettings` object:

```
covariateCohorts <- tibble(
  cohortId = 2,
  cohortName = "Type 2 diabetes"
)

covariateSettings <- createCohortBasedCovariateSettings(
  analysisId = 999,
  covariateCohorts = covariateCohorts,
  valueType = "binary",
  startDay = -365,
  endDay = 0
)
```

Here we first create a tibble containing two columns. The `cohortId` column lists the cohort definition IDs used in our cohort table. The `cohortName` column will be used to create the covariate names. In this case, we only have one cohort in this table, but there could be many. A separate covariate will be created for each cohort.

We then specify an analysis which we assign a unique ID (between 1 and 999). We specify we want to create binary covariates, meaning the covariate will have value = 1 if the cohort is found during the window, and value = 0 if not found. (Because we typically use sparse matrices to represent our covariates, entries with value = 0 will not be included). Alternatively, we could have set `valueType = "count"`, in which case the covariate value would be the number of times the cohort was observed in the time window. Finally, we specify the covariate capture window spans the 365 days before (and including) entry into the main (diclofenac) cohort.

We could also specify the cohort database schema and table where the cohorts for constructing the covariates can be found. However, because both the main cohorts and covariate cohorts are in the same table this is not necessary.

We can now construct the covariates:

```
covariateData <- getDbCovariateData(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = cdmDatabaseSchema,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = cohortTable,
  cohortId = 1,
  rowIdField = "subject_id",
```

```
  covariateSettings = covariateSettings
)
summary(covariateData)
```

```
covariateData$covariateRef
```

In this case we will have only one covariate, diabetes in the year before index. In most cases, we will want our custom covariates in addition to the default covariates. We can do this by creating a list of covariate settings:

```
covariateSettings1 <- createCovariateSettings(
  useDemographicsGender = TRUE,
  useDemographicsAgeGroup = TRUE,
  useDemographicsRace = TRUE,
  useDemographicsEthnicity = TRUE,
  useDemographicsIndexYear = TRUE,
  useDemographicsIndexMonth = TRUE
)

covariateCohorts <- tibble(
  cohortId = 2,
  cohortName = "Type 2 diabetes"
)

covariateSettings2 <- createCohortBasedCovariateSettings(
  analysisId = 999,
  covariateCohorts = covariateCohorts,
  valueType = "binary",
  startDay = -365,
  endDay = 0
)

covariateSettingsList <- list(covariateSettings1, covariateSettings2)

covariateData <- getDbCovariateData(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = cdmDatabaseSchema,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = cohortTable,
  cohortId = 1,
  rowIdField = "subject_id",
  covariateSettings = covariateSettingsList,
  aggregated = TRUE
)
summary(covariateData)
```

In this example both demographic covariates and our diabetes covariate were generated. Note that, for illustration purposes, here we opted for aggregated covariates.