Creating covariates using cohort attributes

Martijn J. Schuemie

2021-12-17

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. However, for some reasons one might need other covariates than those included in the default set.

In the vignette called `Creating custom covariate builders` a process is described for creating custom covariate builders that create covariates on the fly and can be re-used across studies. In contrast, this vignette describes a process that is probably simpler to use, making use of the `cohort_attribute` table in the common data model. It assumes the user has already generated the covariates and loaded them in this table. Below we show a simple example of how this can be done.

2 Overview

To construct custom covariates, the following steps must be taken:

1. Populate a table with the same structure as the `cohort_attribute` table in the common data model.
2. Populate a table with the same structure as the `attribute_definition` table in the common data model.
3. Use the `createCohortAttrCovariateSettings` function to create a `covariateSettings` object pointing to the two tables mentioned in the previous steps.

2.1 Populating the cohort_attribute and attribute_definition tables

The `cohort_attribute` should specify the values of the covariates for every person-`cohort_start_date` combination in the cohort(s) of interest. It should at least have the following fields:

- `cohort_definition_id`, A key to link to the cohort table. Note that this will be come the covariate ID, so you should take care that these IDs do not overlap with IDs of other covariate builders that may be used as well.
- 'subject_id', A key to link to the cohort table.
- 'cohort_start_date', A key to link to the cohort table.
- 'attribute_definition_id', An foreign key linking to the attribute definition table.
- 'value_as_number', A real number.

The first three fields act as a combined key to link the record to an entry in the `cohort` table.

Note that records with zero values can be omitted.

The `attribute_definition` table defines the attributes. It should have at least the following fields:

- 'attribute_definition_id', A unique identifier of type integer.
- 'attribute_name', A short description of the attribute.

The first field links to the `cohort_attribute` table, the second field is used as the covariate name.

3 Example

3.1 Creating the cohort attributes and attributes definitions

In this example we will create a single covariate: the length of observation (LOO) prior to the `cohort_start_date`. We use the following SQL to construct a `cohort_attribute` table and a `attribute_definition` table:

```
/**********************************
File LengthOfObsCohortAttr.sql
**********************************/
IF OBJECT_ID('@cohort_database_schema.@cohort_attribute_table', 'U') IS NOT NULL
    DROP TABLE @cohort_database_schema.@cohort_attribute_table;

IF OBJECT_ID('@cohort_database_schema.@attribute_definition_table', 'U') IS NOT NULL
    DROP TABLE @cohort_database_schema.@attribute_definition_table;

SELECT cohort_definition_id,
    subject_id,
    cohort_start_date,
    1 AS attribute_definition_id,
    DATEDIFF(DAY, observation_period_start_date, cohort_start_date) AS value_as_number
INTO @cohort_database_schema.@cohort_attribute_table
FROM @cohort_database_schema.@cohort_table cohort
INNER JOIN @cdm_database_schema.observation_period op
    ON op.person_id = cohort.subject_id
WHERE cohort_start_date >= observation_period_start_date
    AND cohort_start_date <= observation_period_end_date
```

```
{@cohort_definition_ids != ''} ? {
    AND cohort_definition_id IN (@cohort_definition_ids)
}
;

SELECT 1 AS attribute_definition_id,
  'Length of observation in days' AS attribute_name
INTO @cohort_database_schema.@attribute_definition_table;
```

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

```
library(SqlRender)
sql <- readSql("LengthOfObsCohortAttr.sql")
sql <- render(sql,
                cdm_database_schema = cdmDatabaseSchema,
                cohort_database_schema = cohortDatabaseSchema,
                cohort_table = "rehospitalization",
                cohort_attribute_table = "loo_cohort_attr",
                attribute_definition_table = "loo_attr_def",
                cohort_definition_ids = c(1,2))
sql <- translate(sql, targetDialect = connectionDetails$dbms)

connection <- connect(connectionDetails)
executeSql(connection, sql)
```

3.2 Using the attributes as covariates

To use the constructed attributes as covariates in a predictive model, we need to create a `covariateSettings` object:

```
looCovSet <- createCohortAttrCovariateSettings(attrDatabaseSchema = cohortDatabaseSchema,
                                               cohortAttrTable = "loo_cohort_attr",
                                               attrDefinitionTable = "loo_attr_def",
                                               includeAttrIds = c(),
                                               isBinary = FALSE,
                                               missingMeansZero = FALSE)
```

Setting `isBinary` and `missingMeansZero` is only necessary if we want to aggregate the covariates. We can then use these settings to fetch the `covariates` object:

```
covariates <- getDbCovariateData(connectionDetails = connectionDetails,
                                 cdmDatabaseSchema = cdmDatabaseSchema,
                                 cohortDatabaseSchema = cohortDatabaseSchema,
                                 cohortTable = "rehospitalization",
                                 cohortId = 1,
                                 covariateSettings = looCovSet)
```

In this case we will have only one covariate for our predictive model, the length of observation. 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:

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

looCovSet <- createCohortAttrCovariateSettings(attrDatabaseSchema = cohortDatabaseSchema,
                                               cohortAttrTable = "loo_cohort_attr",
                                               attrDefinitionTable = "loo_attr_def",
                                               includeAttrIds = c())

covariateSettingsList <- list(covariateSettings, looCovSet)

covariates <- getDbCovariateData(connectionDetails = connectionDetails,
                                 cdmDatabaseSchema = cdmDatabaseSchema,
                                 cohortDatabaseSchema = resultsDatabaseSchema,
                                 cohortTable = "rehospitalization",
                                 cohortId = 1,
                                 covariateSettings = covariateSettingsList)
```

In this example both demographic covariates and our length of observation covariate will be generated and can be used in our predictive model.