Creating custom covariate builders (Korean)

Jeon Ga Bin & Martijn J. Schuemie

2024-04-18

1 서론

이 설명서는 사용자가 이미 특징추출 패키지을 사용하는 것에 익숙하다고 가정한다.

특징추출 패키지는 condition_occurrence 테이블에서 발견된 각 조건에대해 공변량 1개와 같은 기본 공변량 집합을 생성 할 수 있다. 그러나 어떠한 이유로든지 기본 집합에 포함된 다른 공변량이 필요할 수도 있다. 때로는 표준 목록에 추가된 새로운 공변량을 쓰는것도 타당할 수 있지만, 다른 때에는 그것들을 안 써야 할 이유가 있을 수도 있다.

특징추출 패키지에는 사용자 정의 공변량 빌더가 있어서 패키지에 포함된 공변량 빌더를 대체하거나 보완하는 메커니즘이 있다. 이 설명서는 그 메커니즘을 설명한다

참고: 사용자 정의 공변량을 추가하는 또 다른 방법은 공통 데이터 모델의 cohort_attribute 테이블을 사용하는 것이다. 이 방법은 creating covariates using cohort attributes라는 설명서에 설명되어 있고, 공변량이 한 번만 필요하거나 고급 R 프로그래밍에 익숙하지 않은 경우 좋은 방법이다. 이 설명서에는 사용자정의 공변량 빌더를 만드는 것을 설명하고 더 복잡하지만 더 많은 연구에서 쉽게 다시 사용할 수 있다.

2 개요

사용자 지정 공변량 빌더를 추가하려면 다음 두가지를 구현해야 한다.

1. 사용자 지정 공변량에 대한 covariateSettings 객체를 생성하는 함수.
2. 공변량 설정을 사용하여 새 공변량을 구성하는 함수.

3 공변량 설정 함수

공변량 설정 함수는 다음 두 가지 요구 사항을 충족하는 개체를 생성해야 한다.

1. 객체의 클래스는 covariateSettings 클래스여야 한다.
2. 객체에는 공변량을 생성하는 함수의 이름을 지정하는 fun 속성이 있어야한다.

3.1 함수 예제

아래에 공변량 설정 함수의 예가 있다.

```r
createLooCovariateSettings <- function(useLengthOfObs = TRUE) {
  covariateSettings <- list(useLengthOfObs = useLengthOfObs)
  attr(covariateSettings, "fun") <- "getDbLooCovariateData"
  class(covariateSettings) <- "covariateSettings"
  return(covariateSettings)
}
```

위의 예시의 함수는 useLengthOfObs라는 인자 하나밖에 없다. 이 인자값은 covariateSettings 객체에 저장되어 있다. 이러한 옵션에 해당하는 공변량을 구성하는 함수의 이름은 getDbLooCovariateData이다.

4 공변량 구성 함수

4.1 함수 입력

공변량 생성 함수는 다음 인자 값들을 받는다:

- connection : DatabaseConnector 패키지의 connect 함수를 사용하여 생선된 스키마가 포함된 서버에 대한 연결이다.
- oracleTempSchema : 오라클에서 임시 테이블을 생성할 수 있는 스키마
- cdmDatabaseSchema : OMOP CDM 인스턴스를 포함하는 데이터베이스 스키마의 이름. SQL SQL 서버에서 이것은 데이터베이스와 스키마를 모두 지정한다(예: cdm_instance.dbo)
- cdmVersion : 사용되는 OMOP CDM 버전 정의: 현재 “4” 및 “5” 지원
- cohortTable : 우리가 공변량을 구성하고자 하는 코호트를 보유하고 있는 테이블의 이름. 이것은 완전히 구체적인 이름이므로 임시테이블의 이름(예: ‘#cohort_table’) 또는 데이터베이스 스키마가 포함된 영구 테이블(예: ’cdm_schema.dbo.cohort)이 될 수 있다.
- cohortIds : 코호트의 코호트 정의 ID. -1로 설정된 경우 코호트 테이블의 모든 항목을 사용하면 된다.
- cdmVersion : 공통 데이터 모델 버전
- rowIdField : 결과 테이블에서 row_id 필드로 사용될 코호트 임시 테이블의 필드 이름이다. 1인당 1개이상의 기간이 있는 경우 특히 유용하다.
- covariateSettings : 공변량 설정 함수에서 생성된 객체
- aggregated : 공변량을 1인당 생성해야 하는가, 아니면 집계해야하는가?

이 함수는 cohort 테이블 인자에 지정된 이름을 가진 테이블이 존재할 것으로 기대할 수 있다. 이 테이블은공변량을 생성하려는 사람과 인덱스날짜를 식별하며 다음 필드를 갖는다(subject_id,cohort_start_date, and cohort_definition_id). 1인당 인덱스 날짜(즉, cohort_start_date)가 두 개 이상일 수 있기 때문에 각 subject_id-cohort_start_date 조합에 대한 고유 식별자가 있는 추가 필드가 포함 될 수 있다. 이 필드의 이름은 rowIdField 인자에 저장된다.

4.2 함수 출력

이 함수는 covariateData 유형의 객체를 반환해야 한다. 이 객체는 다음 멤버 목록을 가진다.

- covariates : 행 ID 당 공변량을 나열하는 ffdf 객체이다. 공간절약을 위해 값 0인 변수를 생략했다. 공변량 객체는 세 개의 열이 있어야한다(rowId,covariateId, and covariateValue)
- covariateRef : 추출된 공변량을 설명하는 ffdf 객체이다. 여기에는 다음과 같은 열이 있어야 한다 (covariateId, covariateName, analysisId, conceptId)
- analysisRef : 함수가 수행한 분석을 설명하는 ffdf 객체이다. 여기에는 다음과 같은 열이 있어야 한다 (analysisId,analysisName,domainIdsta,startDay,endDay,isBinary,missingMeansZero)
- metaData : covariateData 객체의 작성방법에 관한 정보를 가지는 객체의 리스트

4.3 함수 예제

```
getDbLooCovariateData <- function(connection,
                                  oracleTempSchema = NULL,
                                  cdmDatabaseSchema,
                                  cohortTable = "#cohort_person",
                                  cohortIds = c(-1),
                                  cdmVersion = "5",
                                  rowIdField = "subject_id",
                                  covariateSettings,
                                  aggregated = FALSE) {
  writeLines("Constructing length of observation covariates")
  if (covariateSettings$useLengthOfObs == FALSE) {
    return(NULL)
  }
  if (aggregated) {
    stop("Aggregation not supported")
  }

  # Some SQL to construct the covariate:
  sql <- paste("SELECT @row_id_field AS row_id, 1 AS covariate_id,",
               "DATEDIFF(DAY, observation_period_start_date, cohort_start_date)",
               "AS covariate_value",
               "FROM @cohort_table c",
               "INNER JOIN @cdm_database_schema.observation_period op",
               "ON op.person_id = c.subject_id",
               "WHERE cohort_start_date >= observation_period_start_date",
               "AND cohort_start_date <= observation_period_end_date",
               "{@cohort_ids != -1} ? {AND cohort_definition_id IN @cohort_ids}")
  sql <- SqlRender::render(sql,
                           cohort_table = cohortTable,
                           cohort_ids = cohortIds,
                           row_id_field = rowIdField,
                           cdm_database_schema = cdmDatabaseSchema)
  sql <- SqlRender::translate(sql, targetDialect = attr(connection, "dbms"))

  # Retrieve the covariate:
  covariates <- DatabaseConnector::querySql.ffdf(connection, sql)

  # Convert colum names to camelCase:
  colnames(covariates) <- SqlRender::snakeCaseToCamelCase(colnames(covariates))

  # Construct covariate reference:
  covariateRef <- data.frame(
    covariateId = 1,
    covariateName = "Length of observation",
    analysisId = 1,
    conceptId = 0
  )
  covariateRef <- ff::as.ffdf(covariateRef)

  # Construct analysis reference:
  analysisRef <- data.frame(
```

```
    analysisId = 1,
    analysisName = "Length of observation",
    domainId = "Demographics",
    startDay = 0,
    endDay = 0,
    isBinary = "N",
    missingMeansZero = "Y"
  )
  analysisRef <- ff::as.ffdf(analysisRef)

  # Construct analysis reference:
  metaData <- list(sql = sql, call = match.call())
  result <- list(
    covariates = covariates,
    covariateRef = covariateRef,
    analysisRef = analysisRef,
    metaData = metaData
  )
  class(result) <- "covariateData"
  return(result)
}
```

이 예제 함수에서 observation_period_start_date와 인덱스 날짜 사이의 일 수인 '관찰 길이'라는 단일 공변량을 구성한다. 우리는 매개 변수화 된 SQL과 SqlRender 패키지를 사용하여 우리가 연결된 데이터베이스에 적절한 SQL문을 생성한다. DatabaseConnector 패키지를 사용하면 결과가 ffdf 객체에 즉시 저장된다. 우리는 공변량 참조 및 분석 참조 객체를 생성하는데, 이 객체는 하나의 행을 가지며 우리의 공변량과 하나의 분석을 지정한다. 그런 다음 covariate, covariateRef 및 analysisRef 객체를 메타 데이터와 함께 단일 결과 객체로 만든다.

5 사용자 지정 공변량 빌더 사용

PatientLevelPrediction 패키지의 사용자 정의 공변량 빌더와 cohortMethod 패키지와 같은 FeatureExtraction 패키지에 의존하는 다른 패키지를 사용할 수 있다. 사용자 정의 공변량 빌더만 사용하려는 경우, 기존 공변량 설정을 자체 공변량 설정으로 간단히 대체할 수있다.

예를들어:

```
looCovSet <- createLooCovariateSettings(useLengthOfObs = TRUE)

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

이 경우 우리는 예측 모델인 관측 길이에 대한 공변량만 가질 것이다. 대부분의 경우, 우리는 기본 공변량 외에 우리의 맞춤 공변량을 원할 것이다. 공변량 설정 목록을 생성하여 이 작업을 수행할 수 있다.

```
covariateSettings <- createCovariateSettings(
  useDemographicsGender = TRUE,
  useDemographicsAgeGroup = TRUE,
```

```
  useDemographicsRace = TRUE,
  useDemographicsEthnicity = TRUE,
  useDemographicsIndexYear = TRUE,
  useDemographicsIndexMonth = TRUE
)

looCovSet <- createLooCovariateSettings(useLengthOfObs = TRUE)

covariateSettingsList <- list(covariateSettings, looCovSet)

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

이 예에서 인구 통계학적 공변량과 관찰 공변량의 길이가 모두 생성되어 예측 모델에 사용될 수 있다.