Using Characterization Package

Jenna Reps

2024-04-03

Contents

1 Introduction

This vignette describes how you can use the Characterization package for various descriptive studies using OMOP CDM data. The Characterization package currently contains three different types of analyses:

- Aggregate Covariates: this returns the mean feature value for a set of features specified by the user for i) the Target cohort population, ii) the Outcome cohort population, iii) the Target population patients who had the outcome during some user specified time-at-risk and iv) the Target population patients who did not have the outcome during some user specified time-at-risk.
- DechallengeRechallenge: this is mainly aimed at investigating whether a drug and event are causally related by seeing whether the drug is stopped close in time to the event occurrence (dechallenge) and then whether the drug is restarted (a rechallenge occurs) and if so, whether the event starts again (a failed rechallenge). In this analysis, the Target cohorts are the drug users of interest and the Outcome cohorts are the medical events you wish to see whether the drug may cause. The user must also specify how close in time a drug must be stopped after the outcome to be considered a dechallenge and how close in time an Outcome must occur after restarting the drug to be considered a failed rechallenge).
- Time-to-event: this returns descriptive results showing the timing between the target cohort and outcome. This can help identify whether the outcome often precedes the target cohort or whether it generally comes after.

2 Setup

In this vignette we will show working examples using the `Eunomia` R package that contains simulated data. Run the following code to install the `Eunomia` R package:

```r
install.packages("remotes")
remotes::install_github("ohdsi/Eunomia")
```

Eunomia can be used to create a temporary SQLITE database with the simulated data. The function `getEunomiaConnectionDetails` creates a SQLITE connection to a temporary location. The function `createCohorts` then populates the temporary SQLITE database with the simulated data ready to be used.

```r
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
Eunomia::createCohorts(connectionDetails = connectionDetails)
```

```
## Connecting using SQLite driver

## Creating cohort: Celecoxib
##   |

## Executing SQL took 0.0165 secs

## Creating cohort: Diclofenac
##   |

## Executing SQL took 0.0149 secs

## Creating cohort: GiBleed
##   |

## Executing SQL took 0.0195 secs

## Creating cohort: NSAIDs
##   |

## Executing SQL took 0.0627 secs

## Cohorts created in table main.cohort

##   cohortId       name
## 1        1  Celecoxib    A simplified cohort definition for new users of celecoxib, designed specifi
## 2        2 Diclofenac    A simplified cohort definition for new users ofdiclofenac, designed specifi
## 3        3    GiBleed A simplified cohort definition for gastrointestinal bleeding, designed specifi
## 4        4     NSAIDs       A simplified cohort definition for new users of NSAIDs, designed specifi
```

We also need to have the Characterization package installed and loaded

```r
remotes::install_github("ohdsi/FeatureExtraction")
remotes::install_github("ohdsi/Characterization")
```

```r
library(Characterization)
library(dplyr)
```

3 Examples

3.1 Aggreagate Covariates

To run an 'Aggregate Covariate' analysis you need to create a setting object using `createAggregateCovariateSettings`. This requires specifying:

- one or more targetIds (these must be pre-generated in a cohort table)
- one or more outcomeIds (these must be pre-generated in a cohort table)
- the covariate settings using `FeatureExtraction::createCovariateSettings` or by creating your own custom feature extraction code.
- the time-at-risk settings
- riskWindowStart
- startAnchor
- riskWindowEnd
- endAnchor

Using the Eunomia data were we previous generated four cohorts, we can use cohort ids 1,2 and 4 as the targetIds and cohort id 3 as the outcomeIds:

```r
exampleTargetIds <- c(1, 2, 4)
exampleOutcomeIds <- 3
```

If we want to get information on the sex assigned at birth, age at index and Charlson Comorbidity index we can create the settings using `FeatureExtraction::createCovariateSettings`:

```r
exampleCovariateSettings <- FeatureExtraction::createCovariateSettings(
  useDemographicsGender = T,
  useDemographicsAge = T,
  useCharlsonIndex = T
)
```

If we want to create the aggregate features for all our target cohorts, our outcome cohort and each target cohort restricted to those with a record of the outcome 1 day after target cohort start date until 365 days after target cohort end date we can run:

```r
exampleAggregateCovariateSettings <- createAggregateCovariateSettings(
  targetIds = exampleTargetIds,
  outcomeIds = exampleOutcomeIds,
  riskWindowStart = 1, startAnchor = "cohort start",
  riskWindowEnd = 365, endAnchor = "cohort start",
  covariateSettings = exampleCovariateSettings
)
```

Next we need to use the `exampleAggregateCovariateSettings` as the settings to `computeAggregateCovariateAnalyses`, we need to use the Eunomia connectionDetails and in Eunomia the OMOP CDM data and cohort table are in the 'main' schema. The cohort table name is 'cohort'. The following code will apply the aggregated covariates analysis using the previously specified settings on the simulated Eunomia data:

```r
agc <- computeAggregateCovariateAnalyses(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "main",
  cdmVersion = 5,
  targetDatabaseSchema = "main",
  targetTable = "cohort",
  aggregateCovariateSettings = exampleAggregateCovariateSettings,
  databaseId = "Eunomia",
  runId = 1
)
```

If you would like to save the results you can use the function `saveAggregateCovariateAnalyses` and this can then be loaded using `loadAggregateCovariateAnalyses`.

The results are Andromeda objects that can we viewed using `dplyr`. There are four tables:

- covariates:

```r
agc$covariates %>%
  collect() %>%
  kableExtra::kbl()
```

databaseId	runId	cohortDefinitionId	covariateId	sumValue	averageValue
Eunomia	1	1	8507001	237	0.4947808
Eunomia	1	1	8532001	242	0.5052192
Eunomia	1	2	8507001	180	0.5070423
Eunomia	1	2	8532001	175	0.4929577
Eunomia	1	3	8507001	57	0.4596774
Eunomia	1	3	8532001	67	0.5403226
Eunomia	1	4	8507001	237	0.4947808
Eunomia	1	4	8532001	242	0.5052192
Eunomia	1	5	8507001	894	0.4966667
Eunomia	1	5	8532001	906	0.5033333
Eunomia	1	6	8507001	395	0.4759036
Eunomia	1	6	8532001	435	0.5240964
Eunomia	1	7	8507001	1289	0.4901141
Eunomia	1	7	8532001	1341	0.5098859
Eunomia	1	8	8507001	180	0.5070423
Eunomia	1	8	8532001	175	0.4929577
Eunomia	1	9	8507001	57	0.4596774
Eunomia	1	9	8532001	67	0.5403226
Eunomia	1	10	8507001	237	0.4947808
Eunomia	1	10	8532001	242	0.5052192
Eunomia	1	11	8507001	180	0.5070423
Eunomia	1	11	8532001	175	0.4929577
Eunomia	1	12	8507001	57	0.4596774
Eunomia	1	12	8532001	67	0.5403226
Eunomia	1	13	8507001	237	0.4947808
Eunomia	1	13	8532001	242	0.5052192
Eunomia	1	14	8507001	914	0.4956616
Eunomia	1	14	8532001	930	0.5043384
Eunomia	1	15	8507001	407	0.4788235
Eunomia	1	15	8532001	443	0.5211765
Eunomia	1	16	8507001	1321	0.4903489
Eunomia	1	16	8532001	1373	0.5096511
Eunomia	1	17	8507001	237	0.4947808
Eunomia	1	17	8532001	242	0.5052192
Eunomia	1	18	8507001	180	0.5070423
Eunomia	1	18	8532001	175	0.4929577
Eunomia	1	19	8507001	57	0.4596774
Eunomia	1	19	8532001	67	0.5403226
Eunomia	1	20	8507001	237	0.4947808
Eunomia	1	20	8532001	242	0.5052192

- covariatesContinuous:

```r
agc$covariatesContinuous %>%
  collect() %>%
  kableExtra::kbl()
```

databaseId	runId	cohortDefinitionId	covariateId	countValue	minValue	maxValue	averageValue	standardDe
Eunomia	1	3	1901	41	0	2	0.4274194	0.4
Eunomia	1	9	1901	41	0	2	0.4274194	0.4
Eunomia	1	12	1901	41	0	2	0.4274194	0.4
Eunomia	1	19	1901	41	0	2	0.4274194	0.4
Eunomia	1	8	1901	275	0	2	0.9549296	0.4
Eunomia	1	11	1901	275	0	2	0.9549296	0.4
Eunomia	1	2	1901	275	0	2	0.9577465	0.4
Eunomia	1	18	1901	275	0	2	0.9577465	0.4
Eunomia	1	6	1901	296	0	3	0.4024096	0.3
Eunomia	1	15	1901	304	0	3	0.4035294	0.3
Eunomia	1	10	1901	316	0	2	0.8183716	0.4
Eunomia	1	13	1901	316	0	2	0.8183716	0.4
Eunomia	1	1	1901	316	0	2	0.8204593	0.4
Eunomia	1	4	1901	316	0	2	0.8204593	0.4
Eunomia	1	17	1901	316	0	2	0.8204593	0.4
Eunomia	1	20	1901	316	0	2	0.8204593	0.4
Eunomia	1	5	1901	935	0	2	0.6144444	0.3
Eunomia	1	14	1901	958	0	2	0.6144252	0.3
Eunomia	1	7	1901	1231	0	3	0.5475285	0.3
Eunomia	1	16	1901	1262	0	3	0.5478842	0.3
Eunomia	1	9	1002	124	32	46	38.8709677	3.4
Eunomia	1	12	1002	124	32	46	38.8709677	3.4
Eunomia	1	3	1002	124	32	47	38.9758065	3.4
Eunomia	1	19	1002	124	32	47	38.9758065	3.4
Eunomia	1	8	1002	355	32	46	38.7746479	3.2
Eunomia	1	11	1002	355	32	46	38.7746479	3.2
Eunomia	1	2	1002	355	32	46	38.9014085	3.2
Eunomia	1	18	1002	355	32	46	38.9014085	3.2
Eunomia	1	10	1002	479	32	46	38.7995825	3.3
Eunomia	1	13	1002	479	32	46	38.7995825	3.3
Eunomia	1	1	1002	479	32	47	38.9206681	3.2
Eunomia	1	4	1002	479	32	47	38.9206681	3.2
Eunomia	1	17	1002	479	32	47	38.9206681	3.2
Eunomia	1	20	1002	479	32	47	38.9206681	3.2
Eunomia	1	6	1002	830	31	46	38.5746988	3.2
Eunomia	1	15	1002	850	31	46	38.5658824	3.2
Eunomia	1	5	1002	1800	31	47	38.6450000	3.3
Eunomia	1	14	1002	1844	31	47	38.6469631	3.3
Eunomia	1	7	1002	2630	31	47	38.6228137	3.3
Eunomia	1	16	1002	2694	31	47	38.6213808	3.3

- covariateRef:

```r
agc$covariateRef %>%
  collect() %>%
  kableExtra::kbl()
```

databaseId	runId	covariateId	covariateName	analysisId	conceptId
Eunomia	1	8507001	gender = MALE	1	8507
Eunomia	1	8532001	gender = FEMALE	1	8532
Eunomia	1	1901	Charlson index - Romano adaptation	901	0
Eunomia	1	1002	age in years	2	0

- analysisRef:

```
agc$analysisRef %>%
  collect() %>%
  kableExtra::kbl()
```

databaseId	runId	analysisId	analysisName	domainId	startDay	endDay	isBinary	missingMeans
Eunomia	1	1	DemographicsGender	Demographics	NA	NA	Y	NA
Eunomia	1	901	CharlsonIndex	Condition	NA	0	N	Y
Eunomia	1	2	DemographicsAge	Demographics	NA	NA	N	Y

3.2 Dechallenge Rechallenge

To run a ‘Dechallenge Rechallenge’ analysis you need to create a setting object using `createDechallengeRechallengeSettings` This requires specifying:

- one or more targetIds (these must be pre-generated in a cohort table)
- one or more outcomeIds (these must be pre-generated in a cohort table)
- dechallengeStopInterval
- dechallengeEvaluationWindow

Using the Eunomia data were we previous generated four cohorts, we can use cohort ids 1,2 and 4 as the targetIds and cohort id 3 as the outcomeIds:

```
exampleTargetIds <- c(1, 2, 4)
exampleOutcomeIds <- 3
```

If we want to create the dechallenge rechallenge for all our target cohorts and our outcome cohort with a 30 day dechallengeStopInterval and 31 day dechallengeEvaluationWindow:

```
exampleDechallengeRechallengeSettings <- createDechallengeRechallengeSettings(
  targetIds = exampleTargetIds,
  outcomeIds = exampleOutcomeIds,
  dechallengeStopInterval = 30,
  dechallengeEvaluationWindow = 31
)
```

We can then run the analysis on the Eunomia data using `computeDechallengeRechallengeAnalyses` and the settings previously specified:

```
dc <- computeDechallengeRechallengeAnalyses(
  connectionDetails = connectionDetails,
  targetDatabaseSchema = "main",
  targetTable = "cohort",
  dechallengeRechallengeSettings = exampleDechallengeRechallengeSettings,
  databaseId = "Eunomia"
)
```

```
## Inputs checked
```

```
## Connecting using SQLite driver
## Computing dechallenge rechallenge results
##   |

## Executing SQL took 0.00734 secs
## Computing dechallenge rechallenge for 3 target ids and 1outcome ids took 0.163 secs
```

If you would like to save the results you can use the function `saveDechallengeRechallengeAnalyses` and this can then be loaded using `loadDechallengeRechallengeAnalyses`.

The results are Andromeda objects that can we viewed using `dplyr`. There is just one table named dechallengeRechallenge:

```
dc$dechallengeRechallenge %>%
  collect() %>%
  kableExtra::kbl()
```

databaseId	dechallengeStopInterval	dechallengeEvaluationWindow	targetCohortDefinitionId	outcomeCohortDefini

Next it is possible to computer and extract the failed rechallenge cases

```
failed <- computeRechallengeFailCaseSeriesAnalyses(
  connectionDetails = connectionDetails,
  targetDatabaseSchema = "main",
  targetTable = "cohort",
  dechallengeRechallengeSettings = exampleDechallengeRechallengeSettings,
  outcomeDatabaseSchema = "main",
  outcomeTable = "cohort",
  databaseId = "Eunomia"
)
```

```
## Inputs checked

## Connecting using SQLite driver
## Computing dechallenge rechallenge results

##   |

## Executing SQL took 0.0731 secs
## Computing dechallenge failed case series for 3 target IDs and 1 outcome IDs took 0.221 secs
```

The results are Andromeda objects that can we viewed using `dplyr`. There is just one table named rechallengeFailCaseSeries:

```
failed$rechallengeFailCaseSeries %>%
  collect() %>%
  kableExtra::kbl()
```

databaseId	dechallengeStopInterval	dechallengeEvaluationWindow	targetCohortDefinitionId	outcomeCohortDefini

3.3 Time to Event

To run a 'Time-to-event' analysis you need to create a setting object using `createTimeToEventSettings`. This requires specifying:

- one or more targetIds (these must be pre-generated in a cohort table)
- one or more outcomeIds (these must be pre-generated in a cohort table)

```
exampleTimeToEventSettings <- createTimeToEventSettings(
  targetIds = exampleTargetIds,
  outcomeIds = exampleOutcomeIds
)
```

We can then run the analysis on the Eunomia data using `computeTimeToEventAnalyses` and the settings previously specified:

```
tte <- computeTimeToEventAnalyses(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "main",
  targetDatabaseSchema = "main",
  targetTable = "cohort",
  timeToEventSettings = exampleTimeToEventSettings,
  databaseId = "Eunomia"
)
```

```
## Connecting using SQLite driver
## Uploading #cohort_settings
##
## Inserting data took 0.00558 secs
## Computing time to event results

##   |
```

If you would like to save the results you can use the function `saveTimeToEventAnalyses` and this can then be loaded using `loadTimeToEventAnalyses`.

The results are Andromeda objects that can we viewed using `dplyr`. There is just one table named timeToEvent:

```
tte$timeToEvent %>%
  collect() %>%
  top_n(10) %>%
  kableExtra::kbl()
```

```
## Selecting by timeScale
```

databaseId	targetCohortDefinitionId	outcomeCohortDefinitionId	outcomeType	targetOutcomeType	timeToEve
Eunomia	1	3	first	After last target end	3
Eunomia	1	3	first	After last target end	6
Eunomia	1	3	first	After last target end	9
Eunomia	2	3	first	After last target end	3
Eunomia	2	3	first	After last target end	6
Eunomia	2	3	first	After last target end	9
Eunomia	4	3	first	After last target end	3
Eunomia	4	3	first	After last target end	6
Eunomia	4	3	first	After last target end	9
Eunomia	1	3	first	After last target end	36
Eunomia	2	3	first	After last target end	36
Eunomia	4	3	first	After last target end	36

3.4 Run Multiple

If you want to run multiple analyses (of the three previously shown) you can use `createCharacterizationSettings`. You need to input a list of each of the settings (or NULL if you do not want to run one type of analysis). To run all the analyses previously shown in one function:

```
characterizationSettings <- createCharacterizationSettings(
  timeToEventSettings = list(
    exampleTimeToEventSettings
  ),
  dechallengeRechallengeSettings = list(
    exampleDechallengeRechallengeSettings
  ),
  aggregateCovariateSettings = list(
    exampleAggregateCovariateSettings
  )
)

# save the settings using
saveCharacterizationSettings(
  settings = characterizationSettings,
  saveDirectory = file.path(tempdir(), "saveSettings")
)

# the settings can be loaded
characterizationSettings <- loadCharacterizationSettings(
  saveDirectory = file.path(tempdir(), "saveSettings")
)

runCharacterizationAnalyses(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "main",
  targetDatabaseSchema = "main",
  targetTable = "cohort",
  outcomeDatabaseSchema = "main",
  outcomeTable = "cohort",
  characterizationSettings = characterizationSettings,
  saveDirectory = file.path(tempdir(), "example"),
  tablePrefix = "c_",
  databaseId = "1"
)
```

This will create an SQLITE database with all the analyses saved into the saveDirectory. You can export the results as csv files using:

```
connectionDetailsT <- DatabaseConnector::createConnectionDetails(
  dbms = "sqlite",
  server = file.path(tempdir(), "example", "sqliteCharacterization", "sqlite.sqlite")
)

exportDatabaseToCsv(
  connectionDetails = connectionDetailsT,
  resultSchema = "main",
  targetDialect = "sqlite",
  tablePrefix = "c_",
  saveDirectory = file.path(tempdir(), "csv")
)
```