Creating Cohort Subset Definitions

James P. Gilbert and Anthony G. Sena

2024-05-28

Contents

1 Introduction

This guide aims to describe the process of cohort subsetting using `CohortGenerator`. The purpose of Cohort subsetting operations is to allow the creation of common operations that can be applied to generated cohorts in order to subset to different operations in a consistent manner.

2 Subset definitions

Subset definitions are named sets of operations that can be applied to a set of one or more cohorts. The current operations that you can apply to cohorts are:

- Limit subsets
- Demographic subsets
- Cohort subsetting

Operations can be sequentially chained within *subset definitions* and all outputs are considered full cohorts that can be passed into to other packages as if they are cohorts designed in packages

2.1 Demographic subset operations

This subsetting process allows you to capture the age, race/ethnicity gender within a cohort as subgroups. For example, "subset cohorts to subjects that are male between the ages 1 and 5 years old".

2.2 Subsetting to other cohorts

This type of operation allows you to subset a cohort to only those subjects included in one or more other cohorts

3 Creating cohort subset definitions

3.1 Defining a subset definition

First get a Cohort definition set:

```
library(CohortGenerator)
# A list of cohort IDs for use in this vignette
cohortIds <- c(1778211, 1778212, 1778213)
# Get the SQL/JSON for the cohorts
cohortDefinitionSet <- ROhdsiWebApi::exportCohortDefinitionSet(
  baseUrl = baseUrl,
  cohortIds = cohortIds
)
```

A definition can include different subset operations - these are applied strictly in order:

```
# Example, we want to have a HTN cohort that starts any time prior to the index start
# and the HTN cohort ends any time after the index start
subsetDef <- createCohortSubsetDefinition(
  name = "Patients in cohort cohort 1778213 with 365 days prior observation",
  definitionId = 1,
  subsetOperators = list(
    # here we are saying 'first subset to only those patients in cohort 1778213'
    createCohortSubset(
      name = "Subset to patients in cohort 1778213",
      # Note that this can be set to any id - if the
      # cohort is empty or doesn't exist this will not error
      cohortIds = 1778213,
      cohortCombinationOperator = "any",
      negate = FALSE,
      startWindow = createSubsetCohortWindow(
        startDay = -9999,
        endDay = 0,
        targetAnchor = "cohortStart"
      ),
      endWindow = createSubsetCohortWindow(
        startDay = 0,
        endDay = 9999,
        targetAnchor = "cohortStart"
      )
    ),

    # Next, subset to only those with 365 days of prior observation
    createLimitSubset(
      name = "Observation of at least 365 days prior",
      priorTime = 365,
      followUpTime = 0,
      limitTo = "all"
    )
```

```
  )
)
```

3.2 Reusing subset operators in multiple definitions

Next we create a similar definition that also subsetOperators the specified cohorts to require patients with specific demographic criteria. We can do that by copying the subset operations from our first definition and modifying them.

```
subsetOperations2 <- subsetDef$subsetOperators

# subset to those between aged 18 an 64
subsetOperations2[[3]] <-
  createDemographicSubset(
    name = "18 - 65",
    ageMin = 18,
    ageMax = 64
  )

subsetDef2 <- createCohortSubsetDefinition(
  name = "Patients in cohort 1778213 with 365 days prior obs, aged 18 - 64",
  definitionId = 2,
  subsetOperators = subsetOperations2
)
```

3.3 Applying subset definitions to a cohort definition set

Next we need to add the subset definitions to the base cohort set. This will automatically add identifiers and OHDSI SQL for the subset cohorts as well as storing references for saving definition sets for re-use.

```
cohortDefinitionSet <- cohortDefinitionSet |>
  addCohortSubsetDefinition(subsetDef)

knitr::kable(cohortDefinitionSet[, names(cohortDefinitionSet)[which(!names(cohortDefinitionSet) %in% c(
```

cohortId	cohortName	atlasId	logicDescription	isSubset	subsetParent	subsetDefinitionId
1778211	celecoxib	1778211		FALSE	1778211	NA
1778212	celecoxibAge40	1778212		FALSE	1778212	NA
1778213	celecoxibAge40Male	1778213		FALSE	1778213	NA
17782111001	celecoxib - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients in cohort 1778213, Observation of at least 365 days prior	NA	NA	TRUE	1778211	1
17782121001	celecoxibAge40 - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients in cohort 1778213, Observation of at least 365 days prior	NA	NA	TRUE	1778212	1
17782131001	celecoxibAge40Male - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients in cohort 1778213, Observation of at least 365 days prior	NA	NA	TRUE	1778213	1

We can also apply a subset definition to only a limited number of target cohorts as follows

```
cohortDefinitionSet <- cohortDefinitionSet |>
  addCohortSubsetDefinition(subsetDef2, targetCohortIds = 1778212)
```

```
knitr::kable(cohortDefinitionSet[, names(cohortDefinitionSet)[which(!names(cohortDefinitionSet) %in% c(
```

cohortId	cohortName	atlasId	logicDescription	sqlDescription	isSubset	subsetParent	subsetDefinitionId
1778211	celecoxib	1778211			FALSE	1778211	NA
1778212	celecoxibAge40	1778212			FALSE	1778212	NA
1778213	celecoxibAge40Male	1778213			FALSE	1778213	NA
1778211001	celecoxib - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients in cohort 1778213, Observation of at least 365 days prior	NA	NA		TRUE	1778211	1
1778212001	celecoxibAge40 - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients in cohort 1778213, Observation of at least 365 days prior	NA	NA		TRUE	1778212	1
1778213001	celecoxibAge40Male - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients in cohort 1778213, Observation of at least 365 days prior	NA	NA		TRUE	1778213	1
1778212002	celecoxibAge40 - Patients in cohort 1778213 with 365 days prior obs, aged 18 - 64 Subset to patients in cohort 1778213, Observation of at least 365 days prior, 18 - 65	NA	NA		TRUE	1778212	2

The `cohortDefinitionSet` data.frame now has some additional columns:

`subsetParent, isSubset, subsetDefinitionId`

`subsetParent` indicates the parent cohort. For standard cohorts this will be their own ID. For out newly defined subsets, this will be the base cohort.

`subsetDefinitionId` displays the id of the subset applied to the cohort.

In addition, the name of the cohort displayed in this table is automatically generated from the base cohort name, the subset name and the names defined for the subset operations applied in the subset definition. As the number of resulting subsets can become very large, it is crucial to choose human interpretable naming conventions. For example, see the name of our first cohort and the resulting name of a child subset:

```
writeLines(c(
  paste("Cohort Id:", cohortDefinitionSet$cohortId[1]),
  paste("Name", cohortDefinitionSet$cohortName[1])
))
```

```
#> Cohort Id: 1778211
#> Name celecoxib
```

```
writeLines(c(
  paste("Cohort Id:", cohortDefinitionSet$cohortId[4]),
  paste("Subset Parent Id:", cohortDefinitionSet$subsetParent[4]),
  paste("Name", cohortDefinitionSet$cohortName[4])
))
```

```
#> Cohort Id: 1778211001
#> Subset Parent Id: 1778211
#> Name celecoxib - Patients in cohort cohort 1778213 with 365 days prior observation Subset to patients
```

Note that when adding a subset definition to a cohort definition set, the target cohort ids e.g (1778211, 1778212) must exist in the `cohortDefinitionSet` and the output ids (`1778211002`, `1778212003`) must be unique. As with all cohorts, any cohorts with these ids will be deleted prior to execution to prevent collisions. Note that the default expression for output cohort ids is `targetId * 1000 + definitionId` this may cause collisions that will cause `addCohortSubsetDefinition` to error. This can be modified by changing

the `identifierExpression` parameter to `createSubsetDefinition`. This expression should be defined to guarantee uniqueness or adding the definition to a cohort definition set will fail.

4 Generating subsets

Executing CohortGenerator, we can now include the subset operations when our cohorts are generated:

```
connectionDetails <- Eunomia::getEunomiaConnectionDetails()
createCohortTables(
  connectionDetails = connectionDetails,
  cohortDatabaseSchema = "main",
  cohortTableNames = getCohortTableNames("my_cohort")
)
# ### As subsets are a big side effect we need to be clear what was generated and have good naming conv
generatedCohorts <- generateCohortSet(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "main",
  cohortDatabaseSchema = "main",
  cohortTableNames = getCohortTableNames("my_cohort"),
  cohortDefinitionSet = cohortDefinitionSet,
  incremental = TRUE,
  incrementalFolder = file.path(someFolder, "RecordKeeping")
)
```

Cohort subset definitions can be run incrementally. In fact, if the base cohort definition changes for any reason, any subsets will automatically be re-executed when calling `generateCohortSet`.

5 Saving and loading subset definitions

5.1 Saving to packages/directories

Saving applied subsets can automatically be added to a project using `saveCohortDefinitionSet`

```
saveCohortDefinitionSet(cohortDefinitionSet,
  subsetJsonFolder = "<path_to_my_subset_definition>"
)
```

loading is also achieved with `getCohortDefinitionSet`

```
cohortDefinitionSet <- getCohortDefinitionSet(
  subsetJsonFolder = "<path_to_my_subset_definition>"
)
```

Any subset definitions should automatically be loaded and applied to the cohort definition set.

5.2 Writing json objects

Subset definitions can be converted to JSON objects as follows:

```
jsonDefinition <- subsetDef$toJSON()
```

For the purpose of writing to disk we recommend the use of `ParallelLogger` for consistency.

```
# Save to a file
ParallelLogger::saveSettingsToJson(subsetDef$toList(), "subsetDefinition1.json")
```