Using the package skeleton for comparative effectiveness studies

Martijn J. Schuemie

2018-08-24

Contents

1 Introduction

This vignette describes how one can use the package skeleton for comparative effect studies to create one's own study package. This skeleton is aimed at comparative effectiveness studies using the `CohortMethod` package. The resulting package can be used to execute the study at any site that has access to an observational database in the Common Data Model. It will perform the following steps:

1. Instantiate all cohorts needed for the study in a study-specific cohort table.
2. Additional cohorts for negative controls will also be created.
3. (optional) Positive controls will be synthesized, based on the negative controls and pre-specified target effect sizes.
4. The main analysis will be executed using the `CohortMethod` package, including fitting of propensity models, adjusting for the propensity scores, and fitting outcome models.
5. Study diagnostics will be generated while staying blinded to the outcome(s) of interest.
6. Once the study design is finalized (based on the study diagnostics), the results for the outcomes of interest will be generated.

The package skeleton currently implements an examplar study, examining the effect of celecoxib versus diclofenac on gastrointestinal (GI) bleeding. If desired (as a test), one can run the package as is. To run the study, simply run the `execute` function in the package. See the R help system for details:

```
library(SkeletonComparativeEffectStudy)
?execute
```

1.1 extras/CodeToRun.R

Note that for debugging purposes the package developer (you) could story the code for running the study package in your environment in the file called `CodeToRun.R` in the `extras` folder.

1.2 extras/PackageMaintenance.R

This file contains other useful code to be used only by the package developer (you), such as code to generate the package manual, and code to insert cohort definitions into the package. All statements in this file assume the current working directory is set to the root of the package.

Below is the list of steps needed to adapt the package skeleton to implement another study:

2 Copy and rename the package

Please copy the `SkeletonComparativeEffectStudy` folder. Choose a name for your study package, and change all references from 'SkeletonComparativeEffectStudy' to your name of choice in the package code. You could use Notepad++'s 'Find in files' option to do this automatically.

Also, the DESCRIPTION file contains general information about the package, including its authors and a description, so don't forget to update that.

3 Inserting the cohorts of interest

We will assume that the cohorts of interest, specifically the target, comparator, and outcome cohorts, have been defined in ATLAS. These cohort definitions will need to be brought into the study package, so they are available when someone at a different site, who may not have access to the same ATLAS instance, wants to run the study.

First, the cohorts that need to be created need to be listed in the file `inst/settings/CohortsToCreate.csv`. This is a comma-separated file with three columns:

- *cohortId*: An integer ID to identify the cohort throughout the study package. This could be the same as the ID used in ATLAS, but doesn't have to be.
- *atlasId*: The integer ID of the cohort definition in ATLAS.
- *name*: A short name for the cohort. This name is used to create the various file names, so do not use special characters etc. in the name.

Next, we can run the following code, which can also be found in `PackageMaintenance.R`. The baseUrl needs to point to the WebAPI where the cohort definitions are located. The package name needs to be changed to the name you selected:

```
# Insert cohort definitions from ATLAS into package ----------------------
OhdsiRTools::insertCohortDefinitionSetInPackage(fileName = "CohortsToCreate.csv",
                                                baseUrl = Sys.getenv("baseUrl"),
                                                insertTableSql = TRUE,
                                                insertCohortCreationR = TRUE,
                                                generateStats = FALSE,
                                                packageName = "SkeletonComparativeEffectStudy")
```

This code will fetch the cohort definitions from the WebAPI instance, and will insert them as json files in the `inst/cohorts` folder. It will also create corresponding sql files in the `inst/sql` folder.

4 Defining negative control outcomes.

We will assume that a set of negative control outcomes have been defined as a set of concept IDs, with one concept ID per negative control. These negative controls need to be specified in a file called `inst/settings/NegativeControls.csv`. This is a comma-separated file with the following columns:

- *targetId*: The ID of the target cohort as defined in the `CohortsToCreate.csv` file.
- *targetName*: A name for the target cohort. This is not used anywhere, it is just there for convencience.
- *comparatorId*: The ID of the comparator cohort as defined in the `CohortsToCreate.csv` file.
- *comparatorName*: A name for the comparator cohort. This is not used anywhere, it is just there for convencience.
- *outcomeId*: The concept ID of the negative control outcome.
- *outcomeName*: A name for the negative control. This is not used anywhere, it is just there for convencience.
- *type*: Can be `Outcome` for negative control outcomes, and `Exposure` for negative control exposures. Currently, only negative control outcomes are supported.

The reason why for each negative control the target and comparator need to be specified is twofold: First, if multiple target-comparator pairs are investigated in a single study, it may be that not all negative control outcomes are applicable to all target-comparators. Second, in the future we will support negative control exposures, which can then be specified in the same file.

We also need logic to convert the concept IDs into cohorts. This logic is defined in the file `inst/sql/NegativeControlOutcomes.sql`. The default logic simply creates a cohort for every occurrence of the concept ID or any of its descendants in the condition era table. If other logic is required the SQL file can be modified. Be sure to use template SQL as expected by the `SqlRender` package.

5 Define the target-comparator-outcomes of interest

In a single study it is possible to examine multiple target-compartor-outcome triplets. These should be specified in the `inst/settings/TcosOfInterest.csv` file, which has one row per unique target-comparator pair, so could include multiple outcomes per row. his is a comma-separated file with the following columns:

- *targetId*: The ID of the target cohort as defined in the `CohortsToCreate.csv` file.
- *targetName*: A name for the target cohort. This name is used to generate output tables and figures, so should be 'pretty'.
- *comparatorId*: The ID of the comparator cohort as defined in the `CohortsToCreate.csv` file.
- *comparatorName*: A name for the comparator cohort. This name is used to generate output tables and figures, so should be 'pretty'.
- *outcomeIds*: The IDs of the outcome cohorts as defined in the `CohortsToCreate.csv` file. Multiple IDs should be separated using semicolons (;).
- *outcomeNames*: The names of the outcome cohorts. Multiple names should be separated using semicolons (;).These name are used to generate output tables and figures, so should be 'pretty'.
- *excludedCovariateConceptIds*: A list of concept IDs to exclude from the covariates. For example, the concept IDs defining the exposures of interest should typically be excluded and should be listed here. Descendant concepts are automaticaly excluded as well. Multiple IDs should be separated using semicolons (;).

6 Define the analyses settings

The `CohortMethod` package allows multiple analyses settings to be specified, as described in the `Multiple analyses using CohortMethod` vignette. For example, we can define a primary analysis (e.g. using propensity

score matching and an on-treatment time-at-risk definition), and we can additionally define sensitivity analyses (e.g. using propensity score stratification and an intent-to-treat time=at=risk definition). Note that `CohortMethod` will execute all analyses on all target-comparator-outcomes of interest.

We can create these settings and store them in our package. You can modify the file `extras/CreateStudyAnalysisDetails.R` to create the desired study design settings, and use the code below to execute this code, which can also be found in `PackageMaintenance.R`.

```
source("extras/CreateStudyAnalysisDetails.R")
createAnalysesDetails("inst/settings/")
```

7 Positive control synthesis

In addition to negative controls, where the true effect size is believed to be a relative risk of 1, we can also include positive controls, where the true effect has a known magnitude unequal to 1. These positive controls can be used together with the negative controls to perform confidence inteval calibration. We can automatically synthesize positive controls by taking negative controls and adding simulated outcomes during the time at risk. To preserve (measured) confounding, these simulated outcomes should be sampled from the baseline probabilities based on the patients' characteristics.

Positive control synthesis is implemented in the `MethodEvaluation` package, and is called from the skeleton package. However, it is necessary to configure it correctly, in the file called `SynthesizePositiveControls.R` file. Most importantly, the `washoutPeriod`, `riskWindowStart`, `riskWindowEnd`, `addExposureDaysToEnd`, and `removePeopleWithPriorOutcomes` arguments need to be specified to match those in the primary analysis. It is also possible to disable positive control sysnthesis altogether by setting the `skipSynthesis` variable to `TRUE`. Note that even if you set `skipSynthesis <- TRUE`, you still need to use `synthesizePositiveControls = TRUE` when running `execute`.