# Running multiple analyses at once using the CohortMethod package

*Martijn J. Schuemie, Marc A. Suchard and Patrick Ryan*

*2015-06-24*

## Contents

**1 Introduction** **1**

**2 General approach** **1**

**3 Specifying hypotheses of interest** **2**

**4 Specifying analyses** **3**

4.1 Comparator and indication selection strategies . . . . . 4

**5 Executing multiple analyses** **5**

5.1 Restarting . . . . . 6

**6 Acknowledgments** **6**

## 1 Introduction

In this vignette we focus on running several different analyses on several drug-comparator-outcome combinations. This can be useful when we want to explore the sensitivity to analyses choices, include negative controls, or run an experiment similar to the OMOP experiment to empirically identify the optimal analysis choices for a particular research question.

This vignette assumes you are already familiar with the `CohortMethod` package and are able to perform single studies. We will walk through all the steps needed to perform an exemplar set of analyses, and we have selected the well-studied topic of X versus Y on z.

## 2 General approach

The general approach to running a set of analyses is that you specify all the function arguments of the functions you would normally call, and create sets of these function arguments. The final outcome models as well as intermediate data objects will all be saved to disk for later extraction.

An analysis will be executed by calling these functions in sequence:

1. `getDbCohortMethodData()`
2. `createPs()` (optional)
3. `trimByPs()` or `trimByPsToEquipoise()` (optional)

4. `matchOnPs()`, `matchOnPsAndCovariates()`, `stratifyByPs()`, or `stratifyByPsAndCovariates()` (optional)
5. `fitOutcomeModel()` (optional)

When you provide several analyses to the `CohortMethod` package, it will determine whether any of the analyses have anything in common, and will take advantage of this fact. For example, if we specify several analyses that only differ in the way the outcome model is fitted, then `CohortMethod` will extract the data and fit the propensity model only once, and re-use this in all the analysis.

The function arguments you need to define have been divided into four groups:

1. **Hypothesis of interest**: arguments that are specific to a hypothesis of interest, in the case of the cohort method this is a combination of drug, comparator, and outcome (and indication).
2. **Analyses**: arguments that are not directly specific to a hypothesis of interest, such as the washout window, whether to include drugs as covariates, etc.
3. Arguments that are the output of a previous function in the `CohortMethod` package, such as the `cohortMethodData` argument of the `createPs` function. These cannot be specified by the user.
4. Arguments that are specific to an environment, such as the connection details for connecting to the server, and the name of the schema holding the CDM data.

There are a three arguments (`exclusionConceptIds`, `excludedCovariateConceptIds`, and `includedCovariateConceptIds` of the `getDbCohortMethodData()` function) that can be argued to be part both of group 1 and 2. For example, our design in general could exclude cancer patients, and we would specify this as part of the analysis using the `exclusionConceptIds` argument. In other situations we might want to exclude people with prior exposure to the target drug or any related drug, and we would also use the `exclusionConceptIds`. These arguments are therefore present in both groups, and when executing the analysis the union of the two lists of concept IDs will be used.

# 3 Specifying hypotheses of interest

The first group of arguments define the drug, comparator, outcome, and optionally the indication in which to nest the study. Here we demonstrate how to create two sets that only differ in terms of the outcome, and combine them in a list:

```
# TODO: create real example
drugComparatorOutcome1 <- createDrugComparatorOutcome(targetDrugConceptId = 755695,
                                                      comparatorDrugConceptId = 739138,
                                                      indicationConceptIds = 439926,
                                                      outcomeConceptId = 194133)

drugComparatorOutcome2 <- createDrugComparatorOutcome(targetDrugConceptId = 755695,
                                                      comparatorDrugConceptId = 739138,
                                                      indicationConceptIds = 439926,
                                                      outcomeConceptId = 123)

drugComparatorOutcomeList <- list(drugComparatorOutcome1, drugComparatorOutcome2)
```

A convenient way to save `drugComparatorOutcomeList` to file is by using the `saveDrugComparatorOutcomeList` function, and we can load it again using the `loadDrugComparatorOutcomeList` function.

# 4 Specifying analyses

The second group of arguments are not specific to a hypothesis of interest, and comprise the majority of arguments. For each function that will be called during the execution of the analyses, a companion function is available that has (almost) the same arguments. For example, for the `trimByPs()` function there is the `createTrimByPsArgs()` function. These companion functions can be used to create the arguments to be used during execution:

```
getDbCmDataArgs <- createGetDbCohortMethodDataArgs(excludeDrugsFromCovariates = TRUE,
                                                   useCovariateDemographics = TRUE,
                                                   useCovariateConditionOccurrence = TRUE,
                                                   useCovariateConditionOccurrence365d = TRUE,
                                                   useCovariateConditionOccurrence30d = TRUE,
                                                   useCovariateConditionOccurrenceInpt180d = TRUE,
                                                   useCovariateConditionEra = TRUE,
                                                   useCovariateConditionEraEver = TRUE,
                                                   useCovariateConditionEraOverlap = TRUE,
                                                   useCovariateConditionGroup = TRUE,
                                                   useCovariateDrugExposure = TRUE,
                                                   useCovariateDrugExposure365d = TRUE,
                                                   useCovariateDrugExposure30d = TRUE,
                                                   useCovariateDrugEra = TRUE,
                                                   useCovariateDrugEra365d = TRUE,
                                                   useCovariateDrugEra30d = TRUE,
                                                   useCovariateDrugEraEver = TRUE,
                                                   useCovariateDrugEraOverlap = TRUE,
                                                   useCovariateDrugGroup = TRUE,
                                                   useCovariateProcedureOccurrence = TRUE,
                                                   useCovariateProcedureOccurrence365d = TRUE,
                                                   useCovariateProcedureOccurrence30d = TRUE,
                                                   useCovariateProcedureGroup = TRUE,
                                                   useCovariateObservation = TRUE,
                                                   useCovariateObservation365d = TRUE,
                                                   useCovariateObservation30d = TRUE,
                                                   useCovariateObservationBelow = TRUE,
                                                   useCovariateObservationAbove = TRUE,
                                                   useCovariateObservationCount365d = TRUE,
                                                   useCovariateConceptCounts = TRUE,
                                                   useCovariateRiskScores = TRUE,
                                                   useCovariateInteractionYear = FALSE,
                                                   useCovariateInteractionMonth = FALSE,
                                                   deleteCovariatesSmallCount = 100)

createPsArgs <- createCreatePsArgs() # Using only defaults
matchOnPsArgs <- createMatchOnPsArgs(maxRatio = 1)
fitOutcomeModelArgs1 <- createFitOutcomeModelArgs(riskWindowStart = 0,
                                                  riskWindowEnd = 365,
                                                  addExposureDaysToEnd = FALSE,
                                                  modelType = "cox",
                                                  stratifiedCox = TRUE,
                                                  useCovariates = TRUE)
```

Any argument that is not explicitly specified by the user will assume the default value specified in the function. We can now combine the arguments for the various functions into a single analysis:

```
cmAnalysis1 <- createCmAnalysis(analysisId = 1,
                                description = "Analysis using fancy outcome model",
                                getDbCohortMethodDataArgs = getDbCmDataArgs,
                                createPs = TRUE,
                                createPsArgs = createPsArgs,
                                matchOnPs = TRUE,
                                matchOnPsArgs = matchOnPsArgs,
                                fitOutcomeModel = TRUE,
                                fitOutcomeModelArgs = fitOutcomeModelArgs1)
```

Note that we have assigned an analysis ID (1) to this set of arguments. We can use this later to link the results back to this specific set of choices. We also include a short description of the analysis.

We can easily create a second analysis, for example by modifying the outcome model, telling `CohortMethod` not to use the extra covariates in the outcome model:

```
fitOutcomeModelArgs2 <- createFitOutcomeModelArgs(riskWindowStart = 0,
                                                  riskWindowEnd = 365,
                                                  addExposureDaysToEnd = FALSE,
                                                  modelType = "cox",
                                                  stratifiedCox = TRUE,
                                                  useCovariates = FALSE)


cmAnalysis2 <- createCmAnalysis(analysisId = 2,
                                description = "Analysis using simple outcome model",
                                getDbCohortMethodDataArgs = getDbCmDataArgs,
                                createPs = TRUE,
                                createPsArgs = createPsArgs,
                                matchOnPs = TRUE,
                                matchOnPsArgs = matchOnPsArgs,
                                fitOutcomeModel = TRUE,
                                fitOutcomeModelArgs = fitOutcomeModelArgs2)
```

These two analyses can be combined in a list:

```
cmAnalysisList <- list(cmAnalysis1, cmAnalysis2)
```

A convenient way to save `cmAnalysisList` to file is by using the `saveCmAnalysisList` function, and we can load it again using the `loadCmAnalysisList` function.

## 4.1 Comparator and indication selection strategies

Often a new-user cohort design is used for comparative effectiveness studies, where the selection of the comparator is part of the hypothesis of interest: 'Does use of drug A lead to an increased risk compared to use of drug B?', where B is the comparator. But sometimes, the design is used for safety assessment: 'Does use of drug A lead to an increased risk?' In this case the comparator is a proxy for the counterfactual of no treatment. For example, we could pick the comparator to be a drug known not to cause the outcome. we can argue that the selection of the comparator then becomes part of the analyses specification, not the hypothesis of interest, and we can have different strategies for selecting a comparator: Do we for instance pick a drug in the same class, or a drug with the same indication?

In the situation where the comparator choice becomes part of the analyses, we can specify multiple comparators per hypothesis of interest by using a list:

```
# TODO: create real example
comparatorIds = list(drugInSameClass = 1234,
                     drugWithSameIndication = 2345)

drugComparatorOutcome <- createDrugComparatorOutcome(targetDrugConceptId = 755695,
                                                     comparatorDrugConceptId = comparatorIds,
                                                     outcomeConceptId = 194133)

drugComparatorOutcomeList <- list(drugComparatorOutcome)
```

When we specify an analysis, we can then refer to one comparator or another:

```
cmAnalysis1 <- createCmAnalysis(analysisId = 1,
                                description = "Analysis using drug in same class as comparator",
                                comparatorType = "drugInSameClass",
                                getDbCohortMethodDataArgs = getDbCmDataArgs,
                                createPs = TRUE,
                                createPsArgs = createPsArgs,
                                matchOnPs = TRUE,
                                matchOnPsArgs = matchOnPsArgs,
                                fitOutcomeModel = TRUE,
                                fitOutcomeModelArgs = fitOutcomeModelArgs1)

cmAnalysis2 <- createCmAnalysis(analysisId = 2,
                                description = "Analysis using drug with same indication as comparator",
                                comparatorType = "drugWithSameIndication",
                                getDbCohortMethodDataArgs = getDbCmDataArgs,
                                createPs = TRUE,
                                createPsArgs = createPsArgs,
                                matchOnPs = TRUE,
                                matchOnPsArgs = matchOnPsArgs,
                                fitOutcomeModel = TRUE,
                                fitOutcomeModelArgs = fitOutcomeModelArgs1)

cmAnalysisList <- list(cmAnalysis1, cmAnalysis2)
```

# 5 Executing multiple analyses

We can now run the analyses against the hypotheses of interest using the `runCohortMethodAnalyses()` function. This function will run all specified analyses against all hypotheses of interest, meaning that the total number of outcome models is `length(cmAnalysisList) * length(drugComparatorOutcomeList)` (if all analyses specify an outcome model should be fitted).

```
runCmAnalyses(connectionDetails = connectionDetails,
              cdmDatabaseSchema = cdmDatabaseSchema,
              exposureDatabaseSchema = cohortDatabaseSchema,
              exposureTable = cohortTable,
              outcomeDatabaseSchema = cohortDatabaseSchema,
              outcomeTable = cohortTable,
              outputFolder = "./CohortMethodOutput",
              cmAnalysisList,
```

```
                drugComparatorOutcomeList,
                getDbCohortMethodDataThreads = 1,
                createPsThreads = 4,
                fitOutcomeModelThreads = 4)
```

In the code above, we provide the arguments for connecting to the database, which schemas and tables to use, as well as the analyses and hypotheses of interest. The `outputFolder` specifies where the outcome models and intermediate files will be written. We also instruct `CohortMethod` to use 4 threads when fitting propensity scores and when fitting the outcome model. Multithreading can significantly reduce execution time, but can require more system resources such as memory.

### 5.1 Restarting

If for some reason the execution was interrupted, you can restart by re-issuing the `runCohortMethodAnalyses()` command. Any intermediate and final products that have already been completed and written to disk will be skipped.

## 6 Acknowledgments

Considerable work has been dedicated to provide the `CohortMethod` package.

```
citation("CohortMethod")
```

```
#>
#> To cite package 'CohortMethod' in publications use:
#>
#>   Martijn J. Schuemie, Marc A. Suchard and Patrick B. Ryan (2015).
#>   CohortMethod: New-user cohort method with large scale propensity
#>   and outcome models. R package version 1.1.0.
#>
#> A BibTeX entry for LaTeX users is
#>
#>   @Manual{,
#>     title = {CohortMethod: New-user cohort method with large scale propensity and outcome models},
#>     author = {Martijn J. Schuemie and Marc A. Suchard and Patrick B. Ryan},
#>     year = {2015},
#>     note = {R package version 1.1.0},
#>   }
#>
#> ATTENTION: This citation information has been auto-generated from
#> the package DESCRIPTION file and may need manual editing, see
#> 'help("citation")'.
```

Further, `CohortMethod` makes extensive use of the `Cyclops` package.

```
citation("Cyclops")
```

```
#>
#> To cite Cyclops in publications use:
```

```
#>
#> Suchard MA, Simpson SE, Zorych I, Ryan P and Madigan D (2013).
#> "Massive parallelization of serial inference algorithms for
#> complex generalized linear models." _ACM Transactions on Modeling
#> and Computer Simulation_, *23*, pp. 10. <URL:
#> http://dl.acm.org/citation.cfm?id=2414791>.
#>
#> A BibTeX entry for LaTeX users is
#>
#>   @Article{,
#>     author = {M. A. Suchard and S. E. Simpson and I. Zorych and P. Ryan and D. Madigan},
#>     title = {Massive parallelization of serial inference algorithms for complex generalized linear m
#>     journal = {ACM Transactions on Modeling and Computer Simulation},
#>     volume = {23},
#>     pages = {10},
#>     year = {2013},
#>     url = {http://dl.acm.org/citation.cfm?id=2414791},
#>   }
```

This work is supported in part through the National Science Foundation grant IIS 1251151.