Creating Analysis Specification

Anthony G. Sena

2023-03-10

Contents

1 Creating an analysis specification

This walk through will show how to use `Strategus` to define an analysis specification on an example study using cohorts from the example problem *What is the risk of gastrointestinal (GI) bleed in new users of celecoxib compared to new users of diclofenac?* as described in the Book Of OHDSI Chapter 12 on Population Level Estimation

1.1 Cohorts for the study

To start, we'll need to define cohorts and negative control outcomes to use in our example analysis specification. We've included the cohorts and negative control outcomes in the `Strategus` package for this example and the code below will load them for use when assembling the analysis specification.

```
library(CohortGenerator)
cohortDefinitionSet <- getCohortDefinitionSet(
  settingsFileName = "testdata/Cohorts.csv",
  jsonFolder = "testdata/cohorts",
  sqlFolder = "testdata/sql",
```

```
  packageName = "Strategus"
)
ncoCohortSet <- readCsv(file = system.file("testdata/negative_controls_concept_set.csv",
  package = "Strategus"
))
```

1.1.1 Cohort Definitions & Negative Control Outcomes

This is the list of cohort definitions we will use when assembling the analysis specification for Strategus.

```
kable(cohortDefinitionSet[, c("cohortId", "cohortName")])
```

And the negative control outcomes when performing empirical calibration.

```
kable(ncoCohortSet)
```

2 Assembling HADES modules

The building blocks of the `Strategus` analysis specification are HADES modules. For purposes of this walk through, a module is a specific analytic task you would like to perform. As shown in the subsequent sections, the high-level pattern for using a module consists of:

1. Download the module's settings function.
2. Create the module specifications using the settings function(s) from the module
3. Compose the analysis pipeline from one or more module settings

2.1 CohortGenerator Module Settings

The following code downloads the settings functions from the `CohortGeneratorModule` which then activates some additional functions we can use for creating the analysis specification. In the analysis specification, we will add the cohort definitions and negative control outcomes to the `sharedResources` section since these elements may be used by any of the HADES modules. To do this, we will use the `createCohortSharedResourceSpecifications` and `createNegativeControlOutcomeCohortSharedResourceSpecificatio` functions respectively. In addition, we will use the `cohortGeneratorModuleSpecifications` function to specify the cohort generation settings.

```
source("https://raw.githubusercontent.com/OHDSI/CohortGeneratorModule/v0.1.0/SettingsFunctions.R")

# Create the cohort definition shared resource element for the analysis specification
cohortDefinitionSharedResource <- createCohortSharedResourceSpecifications(
  cohortDefinitionSet = cohortDefinitionSet
)

# Create the negative control outcome shared resource element for the analysis specification
ncoSharedResource <- createNegativeControlOutcomeCohortSharedResourceSpecifications(
  negativeControlOutcomeCohortSet = ncoCohortSet,
  occurrenceType = "all",
  detectOnDescendants = TRUE
)
```

```
# Create the module specification
cohortGeneratorModuleSpecifications <- createCohortGeneratorModuleSpecifications(
  incremental = TRUE,
  generateStats = TRUE
)
```

2.2 CohortDiagnostics Module Settings

The following code creates the `cohortDiagnosticsModuleSpecifications` to run cohort diagnostics on the cohorts in the study.

```
source("https://raw.githubusercontent.com/OHDSI/CohortDiagnosticsModule/v0.0.7/SettingsFunctions.R")
cohortDiagnosticsModuleSpecifications <- createCohortDiagnosticsModuleSpecifications(
  runInclusionStatistics = TRUE,
  runIncludedSourceConcepts = TRUE,
  runOrphanConcepts = TRUE,
  runTimeSeries = FALSE,
  runVisitContext = TRUE,
  runBreakdownIndexEvents = TRUE,
  runIncidenceRate = TRUE,
  runCohortRelationship = TRUE,
  runTemporalCohortCharacterization = TRUE,
  incremental = FALSE
)
```

2.3 CohortIncidence Module Settings

The following code creates the `cohortIncidenceModuleSpecifications` to perform an incidence rate analysis for the target cohorts and outcome in this study.

```
source("https://raw.githubusercontent.com/OHDSI/CohortIncidenceModule/v0.0.6/SettingsFunctions.R")
library(CohortIncidence)
targets <- list(
  createCohortRef(id = 1, name = "Celecoxib"),
  createCohortRef(id = 2, name = "Diclofenac"),
  createCohortRef(id = 4, name = "Celecoxib Age >= 30"),
  createCohortRef(id = 5, name = "Diclofenac Age >= 30")
)
outcomes <- list(createOutcomeDef(id = 1, name = "GI bleed", cohortId = 3, cleanWindow = 9999))

tars <- list(
  createTimeAtRiskDef(id = 1, startWith = "start", endWith = "end"),
  createTimeAtRiskDef(id = 2, startWith = "start", endWith = "start", endOffset = 365)
)
analysis1 <- createIncidenceAnalysis(
  targets = c(1, 2, 4, 5),
  outcomes = c(1),
  tars = c(1, 2)
)

irDesign <- createIncidenceDesign(
```

```
  targetDefs = targets,
  outcomeDefs = outcomes,
  tars = tars,
  analysisList = list(analysis1),
  strataSettings = createStrataSettings(
    byYear = TRUE,
    byGender = TRUE
  )
)

cohortIncidenceModuleSpecifications <- createCohortIncidenceModuleSpecifications(
  irDesign = irDesign$toList()
)
```

2.4 Characterization Module Settings

The following code creates the `characterizationModuleSpecifications` to perform an characterization analysis for the target cohorts and outcome in this study.

```
source("https://raw.githubusercontent.com/OHDSI/CharacterizationModule/v0.2.3/SettingsFunctions.R")
characterizationModuleSpecifications <- createCharacterizationModuleSpecifications(
  targetIds = c(1, 2),
  outcomeIds = 3,
  covariateSettings = FeatureExtraction::createDefaultCovariateSettings(),
  dechallengeStopInterval = 30,
  dechallengeEvaluationWindow = 30,
  timeAtRisk = data.frame(
    riskWindowStart = c(1, 1),
    startAnchor = c("cohort start", "cohort start"),
    riskWindowEnd = c(0, 365),
    endAnchor = c("cohort end", "cohort end")
  )
)
```

2.5 CohortMethod Module Settings

The following code creates the `cohortMethodModuleSpecifications` to perform a comparative cohort analysis for this study.

```
library(CohortMethod)
source("https://raw.githubusercontent.com/OHDSI/CohortMethodModule/v0.1.0/SettingsFunctions.R")
negativeControlOutcomes <- lapply(
  X = ncoCohortSet$cohortId,
  FUN = createOutcome,
  outcomeOfInterest = FALSE,
  trueEffectSize = 1,
  priorOutcomeLookback = 30
)

outcomesOfInterest <- lapply(
  X = 3,
```

```
  FUN = createOutcome,
  outcomeOfInterest = TRUE
)

outcomes <- append(
  negativeControlOutcomes,
  outcomesOfInterest
)

tcos1 <- CohortMethod::createTargetComparatorOutcomes(
  targetId = 1,
  comparatorId = 2,
  outcomes = outcomes,
  excludedCovariateConceptIds = c(1118084, 1124300)
)
tcos2 <- CohortMethod::createTargetComparatorOutcomes(
  targetId = 4,
  comparatorId = 5,
  outcomes = outcomes,
  excludedCovariateConceptIds = c(1118084, 1124300)
)

targetComparatorOutcomesList <- list(tcos1, tcos2)

covarSettings <- FeatureExtraction::createDefaultCovariateSettings(addDescendantsToExclude = TRUE)

getDbCmDataArgs <- CohortMethod::createGetDbCohortMethodDataArgs(
  washoutPeriod = 183,
  firstExposureOnly = TRUE,
  removeDuplicateSubjects = "remove all",
  maxCohortSize = 100000,
  covariateSettings = covarSettings
)

createStudyPopArgs <- CohortMethod::createCreateStudyPopulationArgs(
  minDaysAtRisk = 1,
  riskWindowStart = 0,
  startAnchor = "cohort start",
  riskWindowEnd = 30,
  endAnchor = "cohort end"
)

matchOnPsArgs <- CohortMethod::createMatchOnPsArgs()
fitOutcomeModelArgs <- CohortMethod::createFitOutcomeModelArgs(modelType = "cox")
createPsArgs <- CohortMethod::createCreatePsArgs(
  stopOnError = FALSE,
  control = Cyclops::createControl(cvRepetitions = 1)
)
computeSharedCovBalArgs <- CohortMethod::createComputeCovariateBalanceArgs()
computeCovBalArgs <- CohortMethod::createComputeCovariateBalanceArgs(
  covariateFilter = FeatureExtraction::getDefaultTable1Specifications()
)
```

```
cmAnalysis1 <- CohortMethod::createCmAnalysis(
  analysisId = 1,
  description = "No matching, simple outcome model",
  getDbCohortMethodDataArgs = getDbCmDataArgs,
  createStudyPopArgs = createStudyPopArgs,
  fitOutcomeModelArgs = fitOutcomeModelArgs
)

cmAnalysis2 <- CohortMethod::createCmAnalysis(
  analysisId = 2,
  description = "Matching on ps and covariates, simple outcomeModel",
  getDbCohortMethodDataArgs = getDbCmDataArgs,
  createStudyPopArgs = createStudyPopArgs,
  createPsArgs = createPsArgs,
  matchOnPsArgs = matchOnPsArgs,
  computeSharedCovariateBalanceArgs = computeSharedCovBalArgs,
  computeCovariateBalanceArgs = computeCovBalArgs,
  fitOutcomeModelArgs = fitOutcomeModelArgs
)

cmAnalysisList <- list(cmAnalysis1, cmAnalysis2)

analysesToExclude <- NULL


cohortMethodModuleSpecifications <- createCohortMethodModuleSpecifications(
  cmAnalysisList = cmAnalysisList,
  targetComparatorOutcomesList = targetComparatorOutcomesList,
  analysesToExclude = analysesToExclude
)
```

2.6 SelfControlledCaseSeries Module Settings

The following code creates the `cohortMethodModuleSpecifications` to perform a comparative cohort analysis for this study.

```
library(SelfControlledCaseSeries)
source(url = "https://raw.githubusercontent.com/OHDSI/SelfControlledCaseSeriesModule/v0.1.2/SettingsFun

# Exposures-outcomes ----------------------------------------------------------
negativeControlOutcomeIds <- ncoCohortSet$cohortId
outcomeOfInterestIds <- c(3)
exposureOfInterestIds <- c(1, 2)

exposuresOutcomeList <- list()
for (exposureOfInterestId in exposureOfInterestIds) {
  for (outcomeOfInterestId in outcomeOfInterestIds) {
    exposuresOutcomeList[[length(exposuresOutcomeList) + 1]] <- createExposuresOutcome(
      outcomeId = outcomeOfInterestId,
      exposures = list(createExposure(exposureId = exposureOfInterestId))
    )
  }
```

```
  for (negativeControlOutcomeId in negativeControlOutcomeIds) {
    exposuresOutcomeList[[length(exposuresOutcomeList) + 1]] <- createExposuresOutcome(
      outcomeId = negativeControlOutcomeId,
      exposures = list(createExposure(exposureId = exposureOfInterestId, trueEffectSize = 1))
    )
  }
}

# Analysis settings ------------------------------------------------------------
getDbSccsDataArgs <- SelfControlledCaseSeries::createGetDbSccsDataArgs(
  studyStartDate = "",
  studyEndDate = "20191130",
  maxCasesPerOutcome = 1e6,
  useNestingCohort = TRUE,
  nestingCohortId = 1,
  deleteCovariatesSmallCount = 0
)

createStudyPopulation6AndOlderArgs <- SelfControlledCaseSeries::createCreateStudyPopulationArgs(
  minAge = 18,
  naivePeriod = 365
)

covarPreExp <- SelfControlledCaseSeries::createEraCovariateSettings(
  label = "Pre-exposure",
  includeEraIds = "exposureId",
  start = -30,
  end = -1,
  endAnchor = "era start"
)

covarExposureOfInt <- SelfControlledCaseSeries::createEraCovariateSettings(
  label = "Main",
  includeEraIds = "exposureId",
  start = 1,
  startAnchor = "era start",
  end = 0,
  endAnchor = "era end",
  profileLikelihood = TRUE,
  exposureOfInterest = TRUE
)

calendarTimeSettings <- SelfControlledCaseSeries::createCalendarTimeCovariateSettings(
  calendarTimeKnots = 5,
  allowRegularization = TRUE,
  computeConfidenceIntervals = FALSE
)

seasonalitySettings <- SelfControlledCaseSeries::createSeasonalityCovariateSettings(
  seasonKnots = 5,
  allowRegularization = TRUE,
  computeConfidenceIntervals = FALSE
)
```

```
createSccsIntervalDataArgs <- SelfControlledCaseSeries::createCreateSccsIntervalDataArgs(
  eraCovariateSettings = list(covarPreExp, covarExposureOfInt),
  seasonalityCovariateSettings = seasonalitySettings,
  calendarTimeCovariateSettings = calendarTimeSettings,
  minCasesForTimeCovariates = 100000
)

fitSccsModelArgs <- SelfControlledCaseSeries::createFitSccsModelArgs(
  control = Cyclops::createControl(
    cvType = "auto",
    selectorType = "byPid",
    startingVariance = 0.1,
    seed = 1,
    resetCoefficients = TRUE,
    noiseLevel = "quiet"
  )
)

sccsAnalysis1 <- SelfControlledCaseSeries::createSccsAnalysis(
  analysisId = 1,
  description = "SCCS age 18-",
  getDbSccsDataArgs = getDbSccsDataArgs,
  createStudyPopulationArgs = createStudyPopulation6AndOlderArgs,
  createIntervalDataArgs = createSccsIntervalDataArgs,
  fitSccsModelArgs = fitSccsModelArgs
)

sccsAnalysisList <- list(sccsAnalysis1)

# SCCS module specs -----------------------------------------------------------
sccsModuleSpecifications <- creatSelfControlledCaseSeriesModuleSpecifications(
  sccsAnalysisList = sccsAnalysisList,
  exposuresOutcomeList = exposuresOutcomeList,
  combineDataFetchAcrossOutcomes = FALSE
)
```

2.7 PatientLevelPrediction Module Settings

The following code creates the `plpModuleSpecifications` to perform a self-controlled case series analysis for this study.

```
source("https://raw.githubusercontent.com/OHDSI/PatientLevelPredictionModule/v0.0.7/SettingsFunctions.R

makeModelDesignSettings <- function(targetId, outcomeId, popSettings, covarSettings) {
  invisible(PatientLevelPrediction::createModelDesign(
    targetId = targetId,
    outcomeId = outcomeId,
    restrictPlpDataSettings = PatientLevelPrediction::createRestrictPlpDataSettings(),
    populationSettings = popSettings,
    covariateSettings = covarSettings,
    preprocessSettings = PatientLevelPrediction::createPreprocessSettings(),
    modelSettings = PatientLevelPrediction::setLassoLogisticRegression(),
```

```
    splitSettings = PatientLevelPrediction::createDefaultSplitSetting(),
    runCovariateSummary = T
  ))
}

plpPopulationSettings <- PatientLevelPrediction::createStudyPopulationSettings(
  startAnchor = "cohort start",
  riskWindowStart = 1,
  endAnchor = "cohort start",
  riskWindowEnd = 365,
  minTimeAtRisk = 1
)
plpCovarSettings <- FeatureExtraction::createDefaultCovariateSettings()

modelDesignList <- list()
for (i in 1:length(exposureOfInterestIds)) {
  for (j in 1:length(outcomeOfInterestIds)) {
    modelDesignList <- append(
      modelDesignList,
      list(
        makeModelDesignSettings(
          targetId = exposureOfInterestIds[i],
          outcomeId = outcomeOfInterestIds[j],
          popSettings = plpPopulationSettings,
          covarSettings = plpCovarSettings
        )
      )
    )
  }
}

plpModuleSpecifications <- createPatientLevelPredictionModuleSpecifications(modelDesignList = modelDesig
```

3 Strategus Analysis Specifications

Finally, we will use the various shared resources and module specifications to construct the full set of analysis specifications and save it to the file system in JSON format.

```
analysisSpecifications <- createEmptyAnalysisSpecificiations() %>%
  addSharedResources(cohortDefinitionSharedResource) %>%
  addSharedResources(ncoSharedResource) %>%
  addModuleSpecifications(cohortGeneratorModuleSpecifications) %>%
  addModuleSpecifications(cohortDiagnosticsModuleSpecifications) %>%
  addModuleSpecifications(cohortIncidenceModuleSpecifications) %>%
  addModuleSpecifications(characterizationModuleSpecifications) %>%
  addModuleSpecifications(cohortMethodModuleSpecifications) %>%
  addModuleSpecifications(sccsModuleSpecifications) %>%
  addModuleSpecifications(plpModuleSpecifications)

ParallelLogger::saveSettingsToJson(analysisSpecifications, file.path(params$analysisSettingsPath, params
```