Creating Analysis Specification

Anthony G. Sena

2023-09-25

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 Setting up your R environment

To start, we must install the HADES libraries in order to create the settings for the analysis specification. The following script will install the necessary HADES packages for this vignette.

```r
# Install correct versions of HADES packages
remotes::install_github("ohdsi/CohortGenerator", ref = "v0.8.0")
remotes::install_github("ohdsi/CohortDiagnostics", ref = "v3.2.3")
remotes::install_github("ohdsi/Characterization", ref = "v0.1.1")
```

```r
remotes::install_github("ohdsi/CohortIncidence", ref = "v3.1.5")
remotes::install_github("ohdsi/CohortMethod", ref = "v5.1.0")
remotes::install_github("ohdsi/SelfControlledCaseSeries", ref = "v4.2.0")
remotes::install_github("ohdsi/PatientLevelPrediction", ref = "v6.3.4")
```

1.2 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.

```r
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"
))
```

1.2.1 Cohort Definitions & Negative Control Outcomes

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

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

cohortId	cohortName
1	Celecoxib
2	Diclofenac
3	GI bleed
4	Celecoxib age 30 or older
5	Diclofenac age 30 or older

And the negative control outcomes when performing empirical calibration.

```r
kable(ncoCohortSet)
```

cohortId	cohortName	outcomeConceptId
1001	Acute pancreatitis	199074
1002	Alcohol abuse	433753
1003	Allergic rhinitis	257007
1004	Amenorrhea	443800
1005	Bipolar disorder	436665
1006	Candida infection of genital region	4084966
1007	Candidiasis of mouth	29735

cohortId	cohortName	outcomeConceptId
1008	Cardiomegaly	314658
1009	Carotid artery obstruction	4288310
1010	Carpal tunnel syndrome	380094
1011	Chronic obstructive lung disease	255573
1012	Conduct disorder	443617
1013	Dysplasia of cervix	192367
1014	Gastrointestinal hemorrhage	192671
1015	Hirsutism	134718
1016	Hydronephrosis	433811
1017	Hypersomnia	438134
1018	Hypoglycemia	24609
1019	Infectious mononucleosis	437784
1020	Ingrowing nail	139099
1021	Injury of hand	80004
1022	Lipoma (clinical)	440358
1023	Lymphadenitis	316084
1024	Metabolic syndrome X	436940
1025	Mitral valve disorder	319843
1026	Obsessive-compulsive disorder	440374
1027	Ovarian failure	193739
1028	Perforation of tympanic membrane	375292
1029	Peripheral venous insufficiency	321596
1030	Posttraumatic stress disorder	436676
1031	Prostatitis	194997
1032	Restless legs	73754
1033	Senile hyperkeratosis	141932
1034	Sensorineural hearing loss	374366
1035	Sleep apnea	313459
1036	Uterine leiomyoma	197236

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.

```r
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.

```r
source("https://raw.githubusercontent.com/OHDSI/CohortDiagnosticsModule/v0.1.0/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.

```r
source("https://raw.githubusercontent.com/OHDSI/CohortIncidenceModule/v0.2.0/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.

```r
source("https://raw.githubusercontent.com/OHDSI/CharacterizationModule/v0.4.0/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.

```r
library(CohortMethod)
source("https://raw.githubusercontent.com/OHDSI/CohortMethodModule/v0.2.0/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("https://raw.githubusercontent.com/OHDSI/SelfControlledCaseSeriesModule/v0.2.0/SettingsFunctions

# Exposures-outcomes ------------------------------------------------------------
negativeControlOutcomeIds <- ncoCohortSet$cohortId
```

```r
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
)
```

```r
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/v0.2.0/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 = modelDesi
```

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) %>%
```

```r
  addModuleSpecifications(cohortGeneratorModuleSpecifications) %>%
  addModuleSpecifications(cohortDiagnosticsModuleSpecifications) %>%
  addModuleSpecifications(cohortIncidenceModuleSpecifications) %>%
  addModuleSpecifications(characterizationModuleSpecifications) %>%
  addModuleSpecifications(cohortMethodModuleSpecifications) %>%
  addModuleSpecifications(sccsModuleSpecifications) %>%
  addModuleSpecifications(plpModuleSpecifications)

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