Package 'Strategus'

December 4, 2024

Type Package

Title Coordinate and Execute OHDSI HADES Modules

Version 1.1.2

Date 2024-12-04

Maintainer Anthony Sena <sena@ohdsi.org>

Description Coordinate and execute large scale analytics using OHDSI Health
Analytics Data-to-Evidence Suite (HADES) (<https://ohdsi.github.io/Hades/>)
modules.

Depends R (>= 4.2.0),
CohortGenerator (>= 0.11.0),
DatabaseConnector (>= 6.2.3),
R6

Imports checkmate,
cli,
digest,
dplyr,
methods,
ParallelLogger (>= 3.1.0),
purrr,
ResultModelManager (>= 0.5.8),
rlang,
SqlRender (>= 1.18.0)

Suggests Characterization,
CirceR,
CohortDiagnostics,
CohortIncidence,
CohortMethod,
Cyclops,
Eunomia,
EvidenceSynthesis,
FeatureExtraction,
fs,
knitr,
PatientLevelPrediction,
readr,
rmarkdown,
RSQLite,
SelfControlledCaseSeries,

testthat (>= 3.0.0),
withr

Remotes ohdsi/Characterization,
ohdsi/CohortDiagnostics,
ohdsi/CohortGenerator,
ohdsi/CohortIncidence,
ohdsi/CohortMethod,
ohdsi/PatientLevelPrediction,
ohdsi/ResultModelManager,
ohdsi/SelfControlledCaseSeries

License Apache License 2.0

VignetteBuilder knitr

URL https://ohdsi.github.io/Strategus, https://github.com/OHDSI/Strategus

BugReports https://github.com/OHDSI/Strategus/issues

NeedsCompilation no

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Encoding UTF-8

Language en-US

Config/testthat/edition 3

Contents

---

addCharacterizationModuleSpecifications
                    Add Characterization module specifications to analysis specifications

---

Description

Add Characterization module specifications to analysis specifications

Usage

```
addCharacterizationModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications
                An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

moduleSpecifications
                Created by the CharacterizationModule$createModuleSpecifications()
                function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addCohortDiagnosticsModuleSpecifications
                    Add Cohort Diagnostics module specifications to analysis specifica-
                    tions

Description

Add Cohort Diagnostics module specifications to analysis specifications

Usage

```
addCohortDiagnosticsModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications

        An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpecificiations](#)

moduleSpecifications

        Created by the [CohortDiagnosticsModule$createModuleSpecifications()](#) function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addCohortGeneratorModuleSpecifications

        Add Cohort Generator module specifications to analysis specifications

---

Description

Add Cohort Generator module specifications to analysis specifications

Usage

```
addCohortGeneratorModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications

        An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpecificiations](#)

moduleSpecifications

        Created by the [CohortGeneratorModule$createModuleSpecifications()](#) function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addCohortIncidenceModuleSpecifications

        Add Cohort Incidence module specifications to analysis specifications

---

Description

Add Cohort Incidence module specifications to analysis specifications

Usage

```
addCohortIncidenceModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications

An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

moduleSpecifications

Created by the CohortIncidenceModule$createModuleSpecifications() function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addCohortMethodeModuleSpecifications

Add Cohort Method module specifications to analysis specifications

---

Description

Add Cohort Method module specifications to analysis specifications

Usage

```
addCohortMethodeModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications

An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

moduleSpecifications

Created by the CohortMethodModule$createModuleSpecifications() function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addEvidenceSynthesisModuleSpecifications

Add Evidence Synthesis module specifications to analysis specifica-tions

---

Description

Add Evidence Synthesis module specifications to analysis specifications

Usage

```
addEvidenceSynthesisModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications

An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

moduleSpecifications

Created by the EvidenceSynthesisModule$createModuleSpecifications()
function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addModuleSpecifications

Add generic module specifications to analysis specifications

Description

Add generic module specifications to analysis specifications

Usage

```
addModuleSpecifications(analysisSpecifications, moduleSpecifications)
```

Arguments

analysisSpecifications

An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

moduleSpecifications

An object of type ModuleSpecifications

Value

Returns the analysisSpecifications object with the module specifications added.

---

addPatientLevelPredictionModuleSpecifications
                    Add Patient Level Prediction module specifications to analysis speci-
                    fications

Description

Add Patient Level Prediction module specifications to analysis specifications

Usage

```
addPatientLevelPredictionModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications
                    An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations
moduleSpecifications
                    Created by the PatientLevelPredictionModule$createModuleSpecifications()
                    function.

Value

Returns the analysisSpecifications object with the module specifications added.

---

addSelfControlledCaseSeriesModuleSpecifications
                    Add Self Controlled Case Series Module module specifications to anal-
                    ysis specifications

Description

Add Self Controlled Case Series Module module specifications to analysis specifications

Usage

```
addSelfControlledCaseSeriesModuleSpecifications(
  analysisSpecifications,
  moduleSpecifications
)
```

Arguments

analysisSpecifications
                    An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations
moduleSpecifications
                    Created by the SelfControlledCaseSeriesModule$createModuleSpecifications()
                    function.

Value

Returns the `analysisSpecifications` object with the module specifications added.

---

addSharedResources   *Add shared resources (i.e. cohorts) to analysis specifications*

Description

Add shared resources (i.e. cohorts) to analysis specifications

Usage

```
addSharedResources(analysisSpecifications, sharedResources)
```

Arguments

analysisSpecifications

                An object of type `AnalysisSpecifications` as created by `createEmptyAnalysisSpecificiations`

sharedResources

                An object of type `SharedResources`.

Value

Returns the `analysisSpecifications` object with the module specifications added.

CharacterizationModule

                Characterize cohorts with the R*hrefhttps://ohdsi.github.io/Characterization/HADES Characterization Package*

Description

Computes cohort characterization information against the OMOP Common Data Model

Super class

[Strategus::StrategusModule](#) -> CharacterizationModule

Public fields

tablePrefix The table prefix to append to the results tables

Methods

Public methods:

- CharacterizationModule$new()
- CharacterizationModule$execute()
- CharacterizationModule$createResultsDataModel()
- CharacterizationModule$getResultsDataModelSpecification()
- CharacterizationModule$uploadResults()
- CharacterizationModule$createModuleSpecifications()
- CharacterizationModule$clone()

Method new(): Initialize the module

Usage:
```
CharacterizationModule$new()
```

Method execute(): Execute characterization

Usage:
```
CharacterizationModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```
Arguments:

connectionDetails An object of class connectionDetails as created by the [DatabaseConnector::createConne](DatabaseConnector::createConnectionDetails())
    function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](createEmptyAnalysisSpecifications())

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](createEmptyAnalysisSpecifications())

executionSettings An object of type ExecutionSettings as created by createCdmExecutionSettings()
    or createResultsExecutionSettings().

Method createResultsDataModel(): Create the results data model for the module

Usage:
```
CharacterizationModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = self$tablePrefix
)
```
Arguments:

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method getResultsDataModelSpecification(): Get the results data model specification for
the module

Usage:
```
CharacterizationModule$getResultsDataModelSpecification(tablePrefix = "")
```
Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method uploadResults(): Upload the results for the module

Usage:
```
CharacterizationModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```
Arguments:

resultsConnectionDetails The connection details to the results database which is an object
   of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
   function.

resultsConnectionDetails The connection details to the results database which is an object
   of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
   function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

resultsDataModelSettings The results data model settings as created using [@seealso [createResultsDataModel](#)

Method createModuleSpecifications(): Creates the CharacterizationModule Specifications

Usage:
```
CharacterizationModule$createModuleSpecifications(
  targetIds,
  outcomeIds,
  outcomeWashoutDays = c(365),
  minPriorObservation = 365,
  dechallengeStopInterval = 30,
  dechallengeEvaluationWindow = 30,
  riskWindowStart = c(1, 1),
  startAnchor = c("cohort start", "cohort start"),
  riskWindowEnd = c(0, 365),
  endAnchor = c("cohort end", "cohort end"),
  minCharacterizationMean = 0.01,
 covariateSettings = FeatureExtraction::createCovariateSettings(useDemographicsGender =
   T, useDemographicsAge = T, useDemographicsAgeGroup = T, useDemographicsRace = T,
  useDemographicsEthnicity = T, useDemographicsIndexYear = T, useDemographicsIndexMonth
  = T, useDemographicsTimeInCohort = T, useDemographicsPriorObservationTime = T,
   useDemographicsPostObservationTime = T, useConditionGroupEraLongTerm = T,
    useDrugGroupEraOverlapping = T, useDrugGroupEraLongTerm = T,
    useProcedureOccurrenceLongTerm = T, useMeasurementLongTerm = T,

    useObservationLongTerm = T, useDeviceExposureLongTerm = T,
    useVisitConceptCountLongTerm = T, useConditionGroupEraShortTerm = T,
    useDrugGroupEraShortTerm = T, useProcedureOccurrenceShortTerm = T,
   useMeasurementShortTerm = T, useObservationShortTerm = T, useDeviceExposureShortTerm
```

```
  = T, useVisitConceptCountShortTerm = T, endDays = 0, longTermStartDays = -365,
    shortTermStartDays = -30),
  caseCovariateSettings =
   Characterization::createDuringCovariateSettings(useConditionGroupEraDuring = T,
   useDrugGroupEraDuring = T, useProcedureOccurrenceDuring = T, useDeviceExposureDuring
   = T, useMeasurementDuring = T, useObservationDuring = T, useVisitConceptCountDuring =
    T),
  casePreTargetDuration = 365,
  casePostOutcomeDuration = 365
)
```

Arguments:

targetIds A vector of cohort IDs to use as the target(s) for the characterization

outcomeIds A vector of cohort IDs to use as the outcome(s) for the characterization

outcomeWashoutDays A vector of integers specifying the washout days for each outcome (same length as the outcomeIds)

minPriorObservation The number of days of minimum observation a patient in the target populations must have

dechallengeStopInterval description

dechallengeEvaluationWindow description

riskWindowStart The number of days after start anchor to start the time-at-risk (can be a vector for multiple TARS)

startAnchor The TAR starts relative to this either cohort start or cohort end (can be a vector for multiple TARS)

riskWindowEnd The number of days after end anchor to end the time-at-risk (can be a vector for multiple TARS)

endAnchor The TAR ends relative to this either cohort start or cohort end (can be a vector for multiple TARS)

minCharacterizationMean The minimum fraction patients in the target have a covariate for it to be included

covariateSettings Covariates for the database, cohort and risk factor characterization

caseCovariateSettings Covariates for the case-series characterization

casePreTargetDuration The number of days before target start to use for case-series

casePostOutcomeDuration The number of days after outcome start to use for case-series

Method clone(): The objects of this class are cloneable with this method.

Usage:

CharacterizationModule$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

CohortDiagnosticsModule

> *Evaluate phenotypes with the* R*hrefhttps://ohdsi.github.io/CohortDiagnostics/HADES CohortDiagnostics Package*

Description

Development and evaluation of phenotype algorithms against the OMOP Common Data Model.

Super class

[Strategus::StrategusModule](#) -> CohortDiagnosticsModule

Public fields

tablePrefix The table prefix to append to results tables

Methods

Public methods:

- [CohortDiagnosticsModule$new()](#)
- [CohortDiagnosticsModule$execute()](#)
- [CohortDiagnosticsModule$createResultsDataModel()](#)
- [CohortDiagnosticsModule$getResultsDataModelSpecification()](#)
- [CohortDiagnosticsModule$uploadResults()](#)
- [CohortDiagnosticsModule$createModuleSpecifications()](#)
- [CohortDiagnosticsModule$validateModuleSpecifications()](#)
- [CohortDiagnosticsModule$clone()](#)

Method new(): Initialize the module

Usage:
```
CohortDiagnosticsModule$new()
```

Method execute(): Executes the CohortDiagnostics package

Usage:
```
CohortDiagnosticsModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```

Arguments:

connectionDetails An object of class connectionDetails as created by the [DatabaseConnector::createConne](#)
     function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

executionSettings An object of type ExecutionSettings as created by [createCdmExecutionSettings()](#)
     or [createResultsExecutionSettings()](#).

Method createResultsDataModel(): Create the results data model for the module

Usage:
```
CohortDiagnosticsModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = self$tablePrefix
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
     of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
     function.

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method getResultsDataModelSpecification(): Get the results data model specification for the module

Usage:

CohortDiagnosticsModule$getResultsDataModelSpecification(tablePrefix = "")

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method uploadResults(): Upload the results for the module

Usage:

```
CohortDiagnosticsModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

resultsDataModelSettings The results data model settings as created using [@seealso [createResultsDataModel](#)

Method createModuleSpecifications(): Creates the CohortDiagnostics Module Specifications

Usage:

```
CohortDiagnosticsModule$createModuleSpecifications(
  cohortIds = NULL,
  runInclusionStatistics = TRUE,
  runIncludedSourceConcepts = TRUE,
  runOrphanConcepts = TRUE,
  runTimeSeries = FALSE,
  runVisitContext = TRUE,
  runBreakdownIndexEvents = TRUE,
  runIncidenceRate = TRUE,
  runCohortRelationship = TRUE,
  runTemporalCohortCharacterization = TRUE,
  temporalCovariateSettings = private$.getDefaultCovariateSettings(),
  minCharacterizationMean = 0.01,
  irWashoutPeriod = 0
)
```

Arguments:

cohortIds A list of cohort IDs to use when running the CohortDiagnostics. Default is NULL which will use all cohorts present in the cohort definition set in the analysis specification

runInclusionStatistics Generate and export statistic on the cohort inclusion rules?

runIncludedSourceConcepts Generate and export the source concepts included in the cohorts?

runOrphanConcepts Generate and export potential orphan concepts?

runTimeSeries Generate and export the time series diagnostics?

runVisitContext Generate and export index-date visit context?

runBreakdownIndexEvents Generate and export the breakdown of index events?

runIncidenceRate Generate and export the cohort incidence rates?

runCohortRelationship Generate and export the cohort relationship? Cohort relationship checks the temporal relationship between two or more cohorts.

runTemporalCohortCharacterization Generate and export the temporal cohort characterization? Only records with values greater than 0.001 are returned.

temporalCovariateSettings Either an object of type covariateSettings as created using one of the createTemporalCovariateSettings function in the FeatureExtraction package, or a list of such objects.

minCharacterizationMean The minimum mean value for characterization output. Values below this will be cut off from output. This will help reduce the file size of the characterization output, but will remove information on covariates that have very low values. The default is 0.001 (i.e. 0.1 percent)

irWashoutPeriod Number of days washout to include in calculation of incidence rates - default is 0

Method validateModuleSpecifications(): Validate the module specifications

Usage:

CohortDiagnosticsModule$validateModuleSpecifications(moduleSpecifications)

Arguments:

moduleSpecifications The CohortIncidence module specifications

Method clone(): The objects of this class are cloneable with this method.

Usage:

CohortDiagnosticsModule$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

CohortGeneratorModule *Generate cohorts with the* R*hrefhttps://ohdsi.github.io/CohortGenerator/HADES CohortGenerator Package*

Description

Generates cohorts against the OMOP Common Data Model

Super class

Strategus::StrategusModule -> CohortGeneratorModule

Public fields

cohortDefinitionSharedResourcesClassName A constant for the name of the cohort definition shared resources section of the analysis specification

negativeControlOutcomeSharedResourcesClassName A constant for the name of the negative control outcome shared resources section of the analysis specification

Methods

Public methods:

- CohortGeneratorModule$new()
- CohortGeneratorModule$execute()
- CohortGeneratorModule$createResultsDataModel()
- CohortGeneratorModule$getResultsDataModelSpecification()
- CohortGeneratorModule$uploadResults()
- CohortGeneratorModule$createModuleSpecifications()
- CohortGeneratorModule$createCohortSharedResourceSpecifications()
- CohortGeneratorModule$createNegativeControlOutcomeCohortSharedResourceSpecifications()
- CohortGeneratorModule$validateModuleSpecifications()
- CohortGeneratorModule$validateCohortSharedResourceSpecifications()
- CohortGeneratorModule$validateNegativeControlOutcomeCohortSharedResourceSpecifications()
- CohortGeneratorModule$clone()

Method new(): Initialize the module

Usage:
```
CohortGeneratorModule$new()
```

Method execute(): Generates the cohorts

Usage:
```
CohortGeneratorModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```
Arguments:

connectionDetails An object of class connectionDetails as created by the DatabaseConnector::createConne function.

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

executionSettings An object of type ExecutionSettings as created by createCdmExecutionSettings() or createResultsExecutionSettings().

Method createResultsDataModel(): Create the results data model for the module

Usage:
```
CohortGeneratorModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = ""
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method getResultsDataModelSpecification(): Get the results data model specification for
the module

Usage:

CohortGeneratorModule$getResultsDataModelSpecification(tablePrefix = "")

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method uploadResults(): Upload the results for the module

Usage:

```
CohortGeneratorModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#)
function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

resultsDataModelSettings The results data model settings as created using [@seealso [createResultsDataModel](#)

Method createModuleSpecifications(): Creates the CohortGenerator Module Specifica-
tions

Usage:

CohortGeneratorModule$createModuleSpecifications(generateStats = TRUE)

Arguments:

generateStats When TRUE, the Circe cohort definition SQL will include steps to compute
inclusion rule statistics.

Method createCohortSharedResourceSpecifications(): Create shared specifications for
the cohort definition set

Usage:

```
CohortGeneratorModule$createCohortSharedResourceSpecifications(
  cohortDefinitionSet
)
```

Arguments:

cohortDefinitionSet The cohort definition set to include in the specification. See the CohortGenerator package for details on how to build this object.

Method createNegativeControlOutcomeCohortSharedResourceSpecifications(): Create shared specifications for the negative control outcomes cohort set

Usage:

```
CohortGeneratorModule$createNegativeControlOutcomeCohortSharedResourceSpecifications(
  negativeControlOutcomeCohortSet,
  occurrenceType,
  detectOnDescendants
)
```

Arguments:

negativeControlOutcomeCohortSet The negative control outcome cohort definition set defines the concepts to use to construct negative control outcome cohorts. See the CohortGenerator package for more details.

occurrenceType Either "first" or "all

detectOnDescendants When TRUE, the concept ID for the negative control will use the concept_ancestor table and will detect descendant concepts when constructing the cohort.

Method validateModuleSpecifications(): Validate the module specifications

Usage:

```
CohortGeneratorModule$validateModuleSpecifications(moduleSpecifications)
```

Arguments:

moduleSpecifications The CohortGenerator module specifications

Method validateCohortSharedResourceSpecifications(): Validate the cohort shared resource specifications

Usage:

```
CohortGeneratorModule$validateCohortSharedResourceSpecifications(
  cohortSharedResourceSpecifications
)
```

Arguments:

cohortSharedResourceSpecifications The cohort shared resource specifications

Method validateNegativeControlOutcomeCohortSharedResourceSpecifications(): Validate the cohort shared resource specifications

Usage:

```
CohortGeneratorModule$validateNegativeControlOutcomeCohortSharedResourceSpecifications(
  negativeControlOutcomeCohortSharedResourceSpecifications
)
```

Arguments:

negativeControlOutcomeCohortSharedResourceSpecifications The cohort shared resource specifications

Method clone(): The objects of this class are cloneable with this method.

Usage:

CohortGeneratorModule$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

CohortIncidenceModule *Compute incidence with the* R*hrefhttps://ohdsi.github.io/CohortIncidence/HADES CohortIncidence Package*

Description

Computes incidence rates for cohorts against the OMOP Common Data Model

Super class

[Strategus::StrategusModule](#) -> CohortIncidenceModule

Public fields

tablePrefix The table prefix to append to results tables

Methods

Public methods:

- [CohortIncidenceModule$new()](#)
- [CohortIncidenceModule$execute()](#)
- [CohortIncidenceModule$createResultsDataModel()](#)
- [CohortIncidenceModule$getResultsDataModelSpecification()](#)
- [CohortIncidenceModule$uploadResults()](#)
- [CohortIncidenceModule$createModuleSpecifications()](#)
- [CohortIncidenceModule$validateModuleSpecifications()](#)
- [CohortIncidenceModule$clone()](#)

Method new(): Initialize the module

Usage:

CohortIncidenceModule$new()

Method execute(): Execute the CohortIncidence package

Usage:

```
CohortIncidenceModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```

Arguments:

connectionDetails An object of class connectionDetails as created by the [DatabaseConnector::createConne](#) function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec]

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec]

executionSettings An object of type ExecutionSettings as created by [createCdmExecutionSettings()] or [createResultsExecutionSettings()].

Method `createResultsDataModel()`: Create the results data model for the module

Usage:
```
CohortIncidenceModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = ""
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()] function.

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()] function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method `getResultsDataModelSpecification()`: Get the results data model specification for the module

Usage:
```
CohortIncidenceModule$getResultsDataModelSpecification(tablePrefix = "")
```

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method `uploadResults()`: Upload the results for the module

Usage:
```
CohortIncidenceModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()] function.

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()] function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec]

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec]

resultsDataModelSettings The results data model settings as created using [@seealso [createResultsDataModel]

Method createModuleSpecifications(): Creates the CohortIncidence Module Specifications

Usage:

CohortIncidenceModule$createModuleSpecifications(irDesign = NULL)

Arguments:

irDesign The incidence rate design created from the CohortIncidence package

Method validateModuleSpecifications(): Validate the module specifications

Usage:

CohortIncidenceModule$validateModuleSpecifications(moduleSpecifications)

Arguments:

moduleSpecifications The CohortIncidence module specifications

Method clone(): The objects of this class are cloneable with this method.

Usage:

CohortIncidenceModule$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

CohortMethodModule	*New-user cohort studies with the* Rhrefhttps://ohdsi.github.io/CohortMethod/HADES *CohortMethod Package*

Description

Module for performing new-user cohort studies against the OMOP Common Data Model

Super class

[Strategus::StrategusModule](#) -> CohortMethodModule

Methods

Public methods:

- [CohortMethodModule$new()](#)
- [CohortMethodModule$execute()](#)
- [CohortMethodModule$createResultsDataModel()](#)
- [CohortMethodModule$getResultsDataModelSpecification()](#)
- [CohortMethodModule$uploadResults()](#)
- [CohortMethodModule$createModuleSpecifications()](#)
- [CohortMethodModule$validateModuleSpecifications()](#)
- [CohortMethodModule$clone()](#)

Method new(): Initialize the module

Usage:

CohortMethodModule$new()

Method `execute()`: Executes the CohortMethod package

Usage:
```
CohortMethodModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```

Arguments:

connectionDetails An object of class connectionDetails as created by the `DatabaseConnector::createConne`
   function.

analysisSpecifications The analysis specifications for the study

executionSettings An object of type ExecutionSettings as created by `createCdmExecutionSettings()`
   or `createResultsExecutionSettings()`.

Method `createResultsDataModel()`: Create the results data model for the module

Usage:
```
CohortMethodModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = ""
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
   of class connectionDetails as created by the `DatabaseConnector::createConnectionDetails()`
   function.

resultsConnectionDetails The connection details to the results database which is an object
   of class connectionDetails as created by the `DatabaseConnector::createConnectionDetails()`
   function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method `getResultsDataModelSpecification()`: Get the results data model specification for
the module

Usage:
```
CohortMethodModule$getResultsDataModelSpecification(tablePrefix = "")
```

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method `uploadResults()`: Upload the results for the module

Usage:
```
CohortMethodModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
function.

resultsConnectionDetails The connection details to the results database which is an object
of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
function.

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

resultsDataModelSettings The results data model settings as created using [@seealso createResultsDataModel

Method createModuleSpecifications(): Creates the CohortMethod Module Specifications

Usage:
```
CohortMethodModule$createModuleSpecifications(
  cmAnalysisList,
  targetComparatorOutcomesList,
  analysesToExclude = NULL,
  refitPsForEveryOutcome = FALSE,
  refitPsForEveryStudyPopulation = TRUE,
  cmDiagnosticThresholds = CohortMethod::createCmDiagnosticThresholds()
)
```

Arguments:

cmAnalysisList A list of objects of type cmAnalysis as created using the 'CohortMethod::createCmAnalysis
function.

targetComparatorOutcomesList A list of objects of type targetComparatorOutcomes as
created using the CohortMethod::createTargetComparatorOutcomes function.

analysesToExclude Analyses to exclude. See the Analyses to Exclude section for details.

refitPsForEveryOutcome Should the propensity model be fitted for every outcome (i.e. after
people who already had the outcome are removed)? If false, a single propensity model will
be fitted, and people who had the outcome previously will be removed afterwards.

refitPsForEveryStudyPopulation Should the propensity model be fitted for every study
population definition? If false, a single propensity model will be fitted, and the study popu-
lation criteria will be applied afterwards.

cmDiagnosticThresholds An object of type CmDiagnosticThresholds as created using CohortMethod::createC

Details: Run a list of analyses for the target-comparator-outcomes of interest. This function
will run all specified analyses against all hypotheses of interest, meaning that the total number of
outcome models is length(cmAnalysisList) * length(targetComparatorOutcomesList)
(if all analyses specify an outcome model should be fitted). When you provide several analyses
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 this function will extract the data and fit the propensity model only once,
and re-use this in all the analysis.

After completion, a tibble containing references to all generated files can be obtained using
the CohortMethod::getFileReference() function. A summary of the analysis results can be
obtained using the CohortMethod::getResultsSummary() function.

Analyses to Exclude:

Normally, runCmAnalyses will run all combinations of target-comparator-outcome-analyses
settings. However, sometimes we may not need all those combinations. Using the analysesToExclude
argument, we can remove certain items from the full matrix. This argument should be a data
frame with at least one of the following columns:

Method validateModuleSpecifications(): Validate the module specifications

Usage:

```
CohortMethodModule$validateModuleSpecifications(moduleSpecifications)
```

Arguments:

moduleSpecifications The CohortMethod module specifications

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
CohortMethodModule$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

---

createCdmExecutionSettings

Create CDM execution settings

Description

Create CDM execution settings

Usage

```
createCdmExecutionSettings(
  workDatabaseSchema,
  cdmDatabaseSchema,
  cohortTableNames = CohortGenerator::getCohortTableNames(cohortTable = "cohort"),
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  workFolder,
  resultsFolder,
  logFileName = file.path(resultsFolder, "strategus-log.txt"),
  minCellCount = 5,
  incremental = TRUE,
  maxCores = parallel::detectCores(),
  modulesToExecute = c()
)
```

Arguments

workDatabaseSchema

A database schema where intermediate data can be stored. The user (as identified in the connection details) will need to have write access to this database schema.

cdmDatabaseSchema

The database schema containing the data in CDM format. The user (as identified in the connection details) will need to have read access to this database schema.

cohortTableNames

An object identifying the various cohort table names that will be created in the workDatabaseSchema. This object can be created using the `CohortGenerator::getCohortTableNa` function.

tempEmulationSchema

Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

workFolder A folder in the local file system where intermediate results can be written.

resultsFolder The root folder holding the study results.

logFileName Logging information from Strategus and all modules will be located in this file. Individual modules will continue to have their own module-specific logs. By default this will be written to the root of the resultsFolder

minCellCount The minimum number of subjects contributing to a count before it can be included in results.

incremental This value will be passed to each module that supports execution in an incremental manner. Modules and their underlying packages may use the workFolder contents to determine their state of execution and attempt to pick up where they left off when this value is set to TRUE.

maxCores The maximum number of processing cores to use for execution. The default is to use all available cores on the machine.

modulesToExecute

(Optional) A vector with the list of modules to execute. When an empty vector/NULL is supplied (default), all modules in the analysis specification are executed.

Value

An object of type ExecutionSettings.

---

createEmptyAnalysisSpecificiations

Create an empty analysis specifications object.

Description

Create an empty analysis specifications object.

Usage

```
createEmptyAnalysisSpecificiations()
```

Value

An object of type AnalysisSpecifications.

---

createResultDataModel   *Create Result Data Model*

Description

Use this at the study design stage to create data models for modules This functions loads modules and executes any custom code to create the results data model in the specified schema in the results database.

Usage

```
createResultDataModel(
  analysisSpecifications,
  resultsDataModelSettings,
  resultsConnectionDetails
)
```

Arguments

analysisSpecifications

>   An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

resultsDataModelSettings

>   The results data model settings as created using [@seealso createResultsDataModelSettings()]

resultsConnectionDetails

>   The connection details to the results database which is an object of class connectionDetails as created by the DatabaseConnector::createConnectionDetails() function.

---

createResultsDataModelSettings
                Create Results Data Model Settings

Description

The results data model settings are used to create the results data model and to upload results.

Usage

```
createResultsDataModelSettings(
  resultsDatabaseSchema,
  resultsFolder,
  logFileName = file.path(resultsFolder, "strategus-results-data-model-log.txt")
)
```

Arguments

resultsDatabaseSchema

>   The schema in the results database that holds the results data model.

resultsFolder   The root folder holding the study results.

logFileName   Log location for data model operations

Value

An object of type `ResultsDataModelSettings`

---

createResultsExecutionSettings

Create Results execution settings

Description

Create Results execution settings

Usage

```
createResultsExecutionSettings(
  resultsDatabaseSchema,
  workFolder,
  resultsFolder,
  logFileName = file.path(resultsFolder, "strategus-log.txt"),
  minCellCount = 5,
  maxCores = parallel::detectCores(),
  modulesToExecute = c()
)
```

Arguments

resultsDatabaseSchema

The schema in the results database that holds the results data model.

workFolder   A folder in the local file system where intermediate results can be written.

resultsFolder   The root folder holding the study results.

logFileName   Logging information from Strategus and all modules will be located in this file. Individual modules will continue to have their own module-specific logs. By default this will be written to the root of the `resultsFolder`

minCellCount   The minimum number of subjects contributing to a count before it can be included in results.

maxCores   The maximum number of processing cores to use for execution. The default is to use all available cores on the machine.

modulesToExecute

(Optional) A vector with the list of modules to execute. When an empty vector/NULL is supplied (default), all modules in the analysis specification are executed.

Value

An object of type `ExecutionSettings`.

EvidenceSynthesisModule

> *Meta-analysis with the* R*hrefhttps://ohdsi.github.io/EvidenceSynthesis/HADES*
> *EvidenceSynthesis Package*

Description

Module for for combining causal effect estimates and study diagnostics across multiple data sites in a distributed study. This includes functions for performing meta-analysis and forest plots

Super class

[Strategus::StrategusModule](#) -> EvidenceSynthesisModule

Methods

Public methods:

- [EvidenceSynthesisModule$new()](#)
- [EvidenceSynthesisModule$execute()](#)
- [EvidenceSynthesisModule$createResultsDataModel()](#)
- [EvidenceSynthesisModule$getResultsDataModelSpecification()](#)
- [EvidenceSynthesisModule$uploadResults()](#)
- [EvidenceSynthesisModule$validateModuleSpecifications()](#)
- [EvidenceSynthesisModule$createEvidenceSynthesisSource()](#)
- [EvidenceSynthesisModule$createRandomEffectsMetaAnalysis()](#)
- [EvidenceSynthesisModule$createFixedEffectsMetaAnalysis()](#)
- [EvidenceSynthesisModule$createBayesianMetaAnalysis()](#)
- [EvidenceSynthesisModule$createEsDiagnosticThresholds()](#)
- [EvidenceSynthesisModule$createModuleSpecifications()](#)
- [EvidenceSynthesisModule$clone()](#)

Method new(): Initialize the module

Usage:
EvidenceSynthesisModule$new()

Method execute(): Executes the EvidenceSynthesis package

Usage:
EvidenceSynthesisModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)

Arguments:

connectionDetails An object of class connectionDetails as created by the [DatabaseConnector::createConne](#) function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

executionSettings An object of type ExecutionSettings as created by createCdmExecutionSettings() or createResultsExecutionSettings().

Method createResultsDataModel(): Create the results data model for the module

Usage:
```
EvidenceSynthesisModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = ""
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the DatabaseConnector::createConnectionDetails() function.

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the DatabaseConnector::createConnectionDetails() function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method getResultsDataModelSpecification(): Get the results data model specification for the module

Usage:
```
EvidenceSynthesisModule$getResultsDataModelSpecification(tablePrefix = "")
```

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method uploadResults(): Upload the results for the module

Usage:
```
EvidenceSynthesisModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the DatabaseConnector::createConnectionDetails() function.

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the DatabaseConnector::createConnectionDetails() function.

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

resultsDataModelSettings The results data model settings as created using [@seealso createResultsDataModel

Method validateModuleSpecifications(): Validate the module specifications

Usage:
```
EvidenceSynthesisModule$validateModuleSpecifications(moduleSpecifications)
```

Arguments:

moduleSpecifications The EvidenceSynthesis module specifications Create an evidence synthesis source

Method `createEvidenceSynthesisSource()`:

Usage:
```
EvidenceSynthesisModule$createEvidenceSynthesisSource(
  sourceMethod = "CohortMethod",
  databaseIds = NULL,
  analysisIds = NULL,
  likelihoodApproximation = "adaptive grid"
)
```

Arguments:

sourceMethod The source method generating the estimates to synthesize. Can be "CohortMethod" or "SelfControlledCaseSeries"

databaseIds The database IDs to include. Use `databaseIds = NULL` to include all database IDs.

analysisIds The source method analysis IDs to include. Use `analysisIds = NULL` to include all analysis IDs.

likelihoodApproximation The type of likelihood approximation. Can be "adaptive grid" or "normal".

Returns: An object of type `EvidenceSynthesisSource`. Create parameters for a random-effects meta-analysis

Method `createRandomEffectsMetaAnalysis()`:

Usage:
```
EvidenceSynthesisModule$createRandomEffectsMetaAnalysis(
  alpha = 0.05,
  evidenceSynthesisAnalysisId = 1,
  evidenceSynthesisDescription = "Random-effects",
  evidenceSynthesisSource = NULL,
  controlType = "outcome"
)
```

Arguments:

alpha The alpha (expected type I error) used for the confidence intervals.

evidenceSynthesisAnalysisId description

evidenceSynthesisDescription description

evidenceSynthesisSource description

controlType description Create a parameter object for the function computeFixedEffectMetaAnalysis

Details: Use DerSimonian-Laird meta-analysis

Method `createFixedEffectsMetaAnalysis()`:

Usage:

```
EvidenceSynthesisModule$createFixedEffectsMetaAnalysis(
  alpha = 0.05,
  evidenceSynthesisAnalysisId = 1,
  evidenceSynthesisDescription = "Fixed-effects",
  evidenceSynthesisSource = NULL,
  controlType = "outcome"
)
```

Arguments:

alpha The alpha (expected type I error) used for the confidence intervals.

evidenceSynthesisAnalysisId description

evidenceSynthesisDescription description

evidenceSynthesisSource description

controlType description Create a parameter object for the function computeBayesianMeta-Analysis

Details: Create an object defining the parameter values.

Method createBayesianMetaAnalysis()**:**

Usage:

```
EvidenceSynthesisModule$createBayesianMetaAnalysis(
  chainLength = 1100000,
  burnIn = 1e+05,
  subSampleFrequency = 100,
  priorSd = c(2, 0.5),
  alpha = 0.05,
  robust = FALSE,
  df = 4,
  seed = 1,
  evidenceSynthesisAnalysisId = 1,
  evidenceSynthesisDescription = "Bayesian random-effects",
  evidenceSynthesisSource = NULL,
  controlType = "outcome"
)
```

Arguments:

chainLength Number of MCMC iterations.

burnIn Number of MCMC iterations to consider as burn in.

subSampleFrequency Subsample frequency for the MCMC.

priorSd A two-dimensional vector with the standard deviation of the prior for mu and tau, respectively.

alpha The alpha (expected type I error) used for the credible intervals.

robust Whether or not to use a t-distribution model; default: FALSE.

df Degrees of freedom for the t-model, only used if robust is TRUE.

seed The seed for the random number generator.

evidenceSynthesisAnalysisId description

evidenceSynthesisDescription description

evidenceSynthesisSource description

controlType description Create EvidenceSynthesis diagnostics thresholds

Details: Create an object defining the parameter values.

Method `createEsDiagnosticThresholds()`: Threshold used to determine if we pass or fail diagnostics.

Usage:

```
EvidenceSynthesisModule$createEsDiagnosticThresholds(
  mdrrThreshold = 10,
  easeThreshold = 0.25,
  i2Threshold = 0.4,
  tauThreshold = log(2)
)
```

Arguments:

mdrrThreshold What is the maximum allowed minimum detectable relative risk (MDRR)?

easeThreshold What is the maximum allowed expected absolute systematic error (EASE).

i2Threshold What is the maximum allowed I^2 (measure of between-database heterogeneity in random-effects models)?

tauThreshold What is the maximum allowed tau (measure of between-database heterogeneity in Bayesian random-effects models)?

Returns: An object of type `EsDiagnosticThresholds`.

Method `createModuleSpecifications()`: Creates the module Specifications

Usage:

```
EvidenceSynthesisModule$createModuleSpecifications(
  evidenceSynthesisAnalysisList,
  esDiagnosticThresholds = self$createEsDiagnosticThresholds()
)
```

Arguments:

evidenceSynthesisAnalysisList A list of objects of type EvidenceSynthesisAnalysis as generated by either the EvidenceSynthesisModule$createFixedEffectsMetaAnalysis() or EvidenceSynthesisModule$createBayesianMetaAnalysis() function.

esDiagnosticThresholds An object of typeEsDiagnosticThresholds as generated by the EvidenceSynthesisModule$createEsDiagnosticThresholds() function.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
EvidenceSynthesisModule$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

---

execute   *Execute analysis specifications.*

Description

Execute analysis specifications.

Usage

```
execute(analysisSpecifications, executionSettings, connectionDetails)
```

Arguments

analysisSpecifications

An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations

executionSettings

An object of type ExecutionSettings as created by createCdmExecutionSettings()
or createResultsExecutionSettings().

connectionDetails

An object of class connectionDetails as created by the DatabaseConnector::createConnection
function.

Value

Returns a list of lists that contains

- moduleName: The name of the module executed

- result: The result of the execution. See purrr::safely for details on this result.

- executionTime: The time for the module to execute

---

getCdmDatabaseMetaData

Gets the metadata for your OMOP CDM Database

Description

This function is used to gather metadata about your OMOP CDM and inspect for informational
purposes. This information will be saved with your results when executing an analysis specification.

Usage

```
getCdmDatabaseMetaData(cdmExecutionSettings, connectionDetails)
```

Arguments

cdmExecutionSettings

An object of type CdmExecutionSettings as created createCdmExecutionSettings().

connectionDetails

An object of class connectionDetails as created by the DatabaseConnector::createConnection
function.

PatientLevelPredictionModule

Patient-level prediction with the R*hrefhttps://ohdsi.github.io/PatientLevelPrediction/HADES PatientLevelPrediction Package*

Description

Module for performing patient-level prediction in an observational database in the OMOP Common Data Model.

Super class

[Strategus::StrategusModule](#) -> PatientLevelPredictionModule

Public fields

tablePrefix The table prefix to append to the results tables

Methods

Public methods:

- [PatientLevelPredictionModule$new()](#)
- [PatientLevelPredictionModule$execute()](#)
- [PatientLevelPredictionModule$createResultsDataModel()](#)
- [PatientLevelPredictionModule$getResultsDataModelSpecification()](#)
- [PatientLevelPredictionModule$uploadResults()](#)
- [PatientLevelPredictionModule$createModuleSpecifications()](#)
- [PatientLevelPredictionModule$validateModuleSpecifications()](#)
- [PatientLevelPredictionModule$clone()](#)

Method new(): Initialize the module

Usage:
```
PatientLevelPredictionModule$new()
```

Method execute(): Executes the PatientLevelPrediction package

Usage:
```
PatientLevelPredictionModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```

Arguments:

connectionDetails An object of class connectionDetails as created by the [DatabaseConnector::createConne](#) function.

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

analysisSpecifications An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpec](#)

executionSettings An object of type ExecutionSettings as created by [createCdmExecutionSettings()](#) or [createResultsExecutionSettings()](#).

Method createResultsDataModel(): Create the results data model for the module

Usage:
```
PatientLevelPredictionModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = self$tablePrefix
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method getResultsDataModelSpecification(): Get the results data model specification for
the module

Usage:
```
PatientLevelPredictionModule$getResultsDataModelSpecification(tablePrefix = "")
```

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method uploadResults(): Upload the results for the module

Usage:
```
PatientLevelPredictionModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

resultsDataModelSettings The results data model settings as created using [@seealso createResultsDataModel

Method createModuleSpecifications(): Creates the PatientLevelPrediction Module Speci-
fications

Usage:
```
PatientLevelPredictionModule$createModuleSpecifications(modelDesignList)
```

Arguments:

modelDesignList A list of model designs created using `PatientLevelPrediction::createModelDesign()`

Method `validateModuleSpecifications()`: Validate the module specifications

Usage:

`PatientLevelPredictionModule$validateModuleSpecifications(moduleSpecifications)`

Arguments:

moduleSpecifications The PatientLevelPrediction module specifications

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`PatientLevelPredictionModule$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

SelfControlledCaseSeriesModule

Self-Controlled Case Series design with the R*hrefhttps://ohdsi.github.io/SelfControlledCaseSeries/HADES SelfControlledCaseSeries Package*

Description

Module for performing Self-Controlled Case Series (SCCS) analyses against the OMOP Common Data Model.

Super class

[Strategus::StrategusModule](#) -> SelfControlledCaseSeriesModule

Public fields

tablePrefix The table prefix for results tables

Methods

Public methods:

- [SelfControlledCaseSeriesModule$new()](#)
- [SelfControlledCaseSeriesModule$execute()](#)
- [SelfControlledCaseSeriesModule$createResultsDataModel()](#)
- [SelfControlledCaseSeriesModule$getResultsDataModelSpecification()](#)
- [SelfControlledCaseSeriesModule$uploadResults()](#)
- [SelfControlledCaseSeriesModule$createModuleSpecifications()](#)
- [SelfControlledCaseSeriesModule$validateModuleSpecifications()](#)
- [SelfControlledCaseSeriesModule$clone()](#)

Method `new()`: Initialize the module

Usage:

```
SelfControlledCaseSeriesModule$new()
```

Method execute(): Executes the SelfControlledCaseSeries package

Usage:

```
SelfControlledCaseSeriesModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```

Arguments:

connectionDetails An object of class connectionDetails as created by the DatabaseConnector::createConne
    function.

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

executionSettings An object of type ExecutionSettings as created by createCdmExecutionSettings()
    or createResultsExecutionSettings().

Method createResultsDataModel(): Create the results data model for the module

Usage:

```
SelfControlledCaseSeriesModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = ""
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method getResultsDataModelSpecification(): Get the results data model specification for
the module

Usage:

```
SelfControlledCaseSeriesModule$getResultsDataModelSpecification(
  tablePrefix = ""
)
```

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method uploadResults(): Upload the results for the module

Usage:

```
SelfControlledCaseSeriesModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

resultsConnectionDetails The connection details to the results database which is an object
    of class connectionDetails as created by the DatabaseConnector::createConnectionDetails()
    function.

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec

resultsDataModelSettings The results data model settings as created using [@seealso createResultsDataModel

Method createModuleSpecifications(): Creates the SelfControlledCaseSeries Module Specifications

Usage:

```
SelfControlledCaseSeriesModule$createModuleSpecifications(
  sccsAnalysisList,
  exposuresOutcomeList,
  analysesToExclude = NULL,
  combineDataFetchAcrossOutcomes = FALSE,
  sccsDiagnosticThresholds = SelfControlledCaseSeries::createSccsDiagnosticThresholds()
)
```

Arguments:

sccsAnalysisList description

exposuresOutcomeList description

analysesToExclude description

combineDataFetchAcrossOutcomes description

sccsDiagnosticThresholds description

Method validateModuleSpecifications(): Validate the module specifications

Usage:

```
SelfControlledCaseSeriesModule$validateModuleSpecifications(
  moduleSpecifications
)
```

Arguments:

moduleSpecifications The SelfControlledCaseSeries module specifications

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
SelfControlledCaseSeriesModule$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

StrategusModule   *StrategusModule defines the base class for each HADES module*

Description

StrategusModule serves as an internal base class that defines the core functions and structure to be inherited and implemented by any specific HADES module. It provides a standardized framework for creating modular components within the Strategus pipeline.

Public fields

moduleName The name of the module taken from the class name. This is set in the constructor of the class.

moduleClassName The class name that identifies the module specifications in the overall analysis specification. This is set in the constructor of the class.

internalModuleSpecificationClassName A constant value. The base class name that identifies a module specification in the analysis specification.

internalSharedResourcesClassName A constant value. The class name that identifies the shared resources section in the overall analysis specification.

Methods

Public methods:

- StrategusModule$new()
- StrategusModule$execute()
- StrategusModule$createResultsDataModel()
- StrategusModule$getResultsDataModelSpecification()
- StrategusModule$uploadResults()
- StrategusModule$createModuleSpecifications()
- StrategusModule$createSharedResourcesSpecifications()
- StrategusModule$validateModuleSpecifications()
- StrategusModule$validateSharedResourcesSpecifications()
- StrategusModule$clone()

Method new(): Initialize the module

Usage:
```
StrategusModule$new()
```

Method execute(): Executes the module

Usage:
```
StrategusModule$execute(
  connectionDetails,
  analysisSpecifications,
  executionSettings
)
```

Arguments:

connectionDetails An object of class connectionDetails as created by the `DatabaseConnector::createConne` function.

analysisSpecifications An object of type AnalysisSpecifications as created by `createEmptyAnalysisSpec`

analysisSpecifications An object of type AnalysisSpecifications as created by `createEmptyAnalysisSpec`

executionSettings An object of type ExecutionSettings as created by `createCdmExecutionSettings()` or `createResultsExecutionSettings()`.

Method `createResultsDataModel()`: Create the results data model for the module

Usage:
```
StrategusModule$createResultsDataModel(
  resultsConnectionDetails,
  resultsDatabaseSchema,
  tablePrefix = ""
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the `DatabaseConnector::createConnectionDetails()` function.

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the `DatabaseConnector::createConnectionDetails()` function.

resultsDatabaseSchema The schema in the results database that holds the results data model.

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method `getResultsDataModelSpecification()`: Get the results data model specification for the module

Usage:
```
StrategusModule$getResultsDataModelSpecification(tablePrefix = "")
```

Arguments:

tablePrefix A prefix to apply to the database table names (optional).

tablePrefix A prefix to apply to the database table names (optional).

Method `uploadResults()`: Upload the results for the module

Usage:
```
StrategusModule$uploadResults(
  resultsConnectionDetails,
  analysisSpecifications,
  resultsDataModelSettings
)
```

Arguments:

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the `DatabaseConnector::createConnectionDetails()` function.

resultsConnectionDetails The connection details to the results database which is an object of class connectionDetails as created by the `DatabaseConnector::createConnectionDetails()` function.

analysisSpecifications An object of type AnalysisSpecifications as created by `createEmptyAnalysisSpec`

analysisSpecifications An object of type AnalysisSpecifications as created by createEmptyAnalysisSpec
resultsDataModelSettings The results data model settings as created using [@seealso createResultsDataModel

Method createModuleSpecifications(): Base function for creating the module settings object. Each module will have its own implementation and this base class method will be used to ensure the class of the specifications is set properly.

Usage:

StrategusModule$createModuleSpecifications(moduleSpecifications)

Arguments:

moduleSpecifications An object of type ModuleSpecifications

moduleSpecifications An object of type ModuleSpecifications

Method createSharedResourcesSpecifications(): Base function for creating the shared resources settings object. Each module will have its own implementation if it needs to create a shared resource.

Usage:

StrategusModule$createSharedResourcesSpecifications(
  className,
  sharedResourcesSpecifications
)

Arguments:

className The class name of the shared resources specifications

sharedResourcesSpecifications The shared resources specifications

Method validateModuleSpecifications(): Base function for validating the module settings object. Each module will have its own implementation and this base class method will be used to ensure the module specifications are valid ahead of execution

Usage:

StrategusModule$validateModuleSpecifications(moduleSpecifications)

Arguments:

moduleSpecifications An object of type ModuleSpecifications

moduleSpecifications An object of type ModuleSpecifications

Method validateSharedResourcesSpecifications(): Base function for validating the shared resources specification settings object. Each module will have its own implementation and this base class method will be used to ensure the module specifications are valid ahead of execution

Usage:

StrategusModule$validateSharedResourcesSpecifications(
  className,
  sharedResourcesSpecifications
)

Arguments:

className The class name of the shared resources specifications

sharedResourcesSpecifications The shared resources specifications

Method clone(): The objects of this class are cloneable with this method.

Usage:

StrategusModule$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

---

uploadResults   *Upload results*

Description

Upload the results for a given analysis

Usage

```
uploadResults(
  analysisSpecifications,
  resultsDataModelSettings,
  resultsConnectionDetails
)
```

Arguments

analysisSpecifications

An object of type AnalysisSpecifications as created by [createEmptyAnalysisSpecificiations](#)

resultsDataModelSettings

The results data model settings as created using [@seealso [createResultsDataModelSettings()](#)]

resultsConnectionDetails

The connection details to the results database which is an object of class connectionDetails as created by the [DatabaseConnector::createConnectionDetails()](#) function.

---

zipResults   *Create a zip file with all study results for sharing with study coordinator*

Description

Create a zip file with all study results for sharing with study coordinator

Usage

```
zipResults(resultsFolder, zipFile)
```

Arguments

resultsFolder   The root folder holding the study results.

zipFile   The path to the zip file to be created.

Details

Creates a .zip file of the .csv files found in the resultsFolder. The resulting .zip file will have relative paths to the root of the resultsFolder which is generally found in executionSettings$resultsFolder.

Value

Does not return anything. Is called for the side-effect of creating the zip file with results.

Index