Package ‘Characterization’

March 4, 2025

Type Package

Title Implement Descriptive Studies Using the Common Data Model

Version 2.1.3

Date 2025-2-26

Maintainer Jenna Reps <`jreps@its.jnj.com`>

Description An end-to-end framework that enables users to implement various descriptive studies for a given set of target and outcome cohorts for data mapped to the Observational Medical Outcomes Partnership Common Data Model.

License Apache License 2.0

URL `https://ohdsi.github.io/Characterization/`,
`https://github.com/OHDSI/Characterization`

BugReports `https://github.com/OHDSI/Characterization/issues`

Depends R (>= 4.0.0)

Imports Andromeda, DatabaseConnector (>= 6.3.1), FeatureExtraction (>= 3.6.0), SqlRender (>= 1.9.0), ParallelLogger (>= 3.0.0), ResultModelManager, checkmate, dplyr, readr, rlang

Suggests devtools, testthat, kableExtra, knitr, markdown, rmarkdown, OhdsiShinyAppBuilder, shiny, withr

NeedsCompilation no

RoxygenNote 7.3.2

Encoding UTF-8

VignetteBuilder knitr

Author Jenna Reps [aut, cre],
Patrick Ryan [aut],
Chris Knoll [aut]

Repository CRAN

Date/Publication 2025-03-04 12:50:02 UTC

Contents

`cleanIncremental` *Removes csv files from folders that have not been marked as completed and removes the record of the execution file*

Description

Removes csv files from folders that have not been marked as completed and removes the record of the execution file

Usage

```
cleanIncremental(executionFolder, ignoreWhenEmpty = FALSE)
```

Arguments

`executionFolder`	The folder that has the execution files
`ignoreWhenEmpty`	When TRUE, if there are no incremental logs then nothing is run

Value

A list with the settings

See Also

Other Incremental: `cleanNonIncremental()`

Examples

```
cleanIncremental(
  file.path(tempdir(), 'incremental'),
  ignoreWhenEmpty = TRUE
)
```

`cleanNonIncremental` *Removes csv files from the execution folder as there should be no csv files when running in non-incremental model*

Description

Removes csv files from the execution folder as there should be no csv files when running in non-incremental model

Usage

```
cleanNonIncremental(executionFolder)
```

Arguments

`executionFolder`	The folder that has the execution files

Value

A list with the settings

See Also

Other Incremental: `cleanIncremental()`

Examples

```
# example code

cleanNonIncremental(file.path(tempdir(), 'incremental'))
```

computeDechallengeRechallengeAnalyses

Compute dechallenge rechallenge study

Description

Compute dechallenge rechallenge study

Usage

```
computeDechallengeRechallengeAnalyses(
  connectionDetails = NULL,
  targetDatabaseSchema,
  targetTable,
  outcomeDatabaseSchema = targetDatabaseSchema,
  outcomeTable = targetTable,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  settings,
  databaseId = "database 1",
  outputFolder,
  minCellCount = 0,
  ...
)
```

Arguments

`connectionDetails`	An object of type ‘connectionDetails‘ as created using the [DatabaseConnector::createConnectionDetails()] function.
`targetDatabaseSchema`	Schema name where your target cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`targetTable`	Name of the target cohort table.
`outcomeDatabaseSchema`	Schema name where your outcome cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`outcomeTable`	Name of the outcome cohort table.
`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
`settings`	The settings for the timeToEvent study
`databaseId`	An identifier for the database (string)
`outputFolder`	A directory to save the results as csv files

`minCellCount`	The minimum cell value to display, values less than this will be replaced by -1
`...`	extra inputs

Value

An `Andromeda::andromeda()` object containing the dechallenge rechallenge results

See Also

Other DechallengeRechallenge: `computeRechallengeFailCaseSeriesAnalyses()`, `createDechallengeRechallengeSe`

Examples

```
conDet <- exampleOmopConnectionDetails()

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

computeDechallengeRechallengeAnalyses(
  connectionDetails = conDet,
  targetDatabaseSchema = 'main',
  targetTable = 'cohort',
  settings = drSet,
  outputFolder = tempdir()
)
```

`computeRechallengeFailCaseSeriesAnalyses`

Compute fine the subjects that fail the dechallenge rechallenge study

---

Description

Compute fine the subjects that fail the dechallenge rechallenge study

Usage

```
computeRechallengeFailCaseSeriesAnalyses(
  connectionDetails = NULL,
  targetDatabaseSchema,
  targetTable,
  outcomeDatabaseSchema = targetDatabaseSchema,
  outcomeTable = targetTable,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  settings,
  databaseId = "database 1",
  showSubjectId = FALSE,
  outputFolder,
  minCellCount = 0,
  ...
)
```

Arguments

`connectionDetails`	An object of type ‘connectionDetails‘ as created using the [DatabaseConnector::createConnectionDetails()] function.
`targetDatabaseSchema`	Schema name where your target cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`targetTable`	Name of the target cohort table.
`outcomeDatabaseSchema`	Schema name where your outcome cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`outcomeTable`	Name of the outcome cohort table.
`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
`settings`	The settings for the timeToEvent study
`databaseId`	An identifier for the database (string)
`showSubjectId`	if F then subject_ids are hidden (recommended if sharing results)
`outputFolder`	A directory to save the results as csv files
`minCellCount`	The minimum cell value to display, values less than this will be replaced by -1
`...`	extra inputs

Value

An `Andromeda::andromeda()` object with the case series details of the failed rechallenge

See Also

Other DechallengeRechallenge: `computeDechallengeRechallengeAnalyses()`, `createDechallengeRechallengeSetti`

Examples

```
conDet <- exampleOmopConnectionDetails()

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)
```

```
computeRechallengeFailCaseSeriesAnalyses(
  connectionDetails = conDet,
  targetDatabaseSchema = 'main',
  targetTable = 'cohort',
  settings = drSet,
  outputFolder = tempdir()
)
```

---

computeTimeToEventAnalyses

Compute time to event study

Description

Compute time to event study

Usage

```
computeTimeToEventAnalyses(
  connectionDetails = NULL,
  targetDatabaseSchema,
  targetTable,
  outcomeDatabaseSchema = targetDatabaseSchema,
  outcomeTable = targetTable,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cdmDatabaseSchema,
  settings,
  databaseId = "database 1",
  outputFolder,
  minCellCount = 0,
  ...
)
```

Arguments

`connectionDetails`	An object of type ‘connectionDetails‘ as created using the [DatabaseConnector::createConnectionDetails()] function.
`targetDatabaseSchema`	Schema name where your target cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example ’scratch.dbo’.
`targetTable`	Name of the target cohort table.
`outcomeDatabaseSchema`	Schema name where your outcome cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example ’scratch.dbo’.

`outcomeTable`	Name of the outcome cohort table.
`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
`cdmDatabaseSchema`	The database schema containing the OMOP CDM data
`settings`	The settings for the timeToEvent study
`databaseId`	An identifier for the database (string)
`outputFolder`	A directory to save the results as csv files
`minCellCount`	The minimum cell value to display, values less than this will be replaced by -1
`...`	extra inputs

Value

An `Andromeda::andromeda()` object containing the time to event results.

See Also

Other TimeToEvent: `createTimeToEventSettings()`

Examples

```
# example code

conDet <- exampleOmopConnectionDetails()

tteSet <- createTimeToEventSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

result <- computeTimeToEventAnalyses(
  connectionDetails = conDet,
  targetDatabaseSchema = 'main',
  targetTable = 'cohort',
  cdmDatabaseSchema = 'main',
  settings = tteSet,
  outputFolder = file.path(tempdir(), 'tte')
)
```

createAggregateCovariateSettings

Create aggregate covariate study settings

Description

Create aggregate covariate study settings

Usage

```
createAggregateCovariateSettings(
  targetIds,
  outcomeIds,
  minPriorObservation = 0,
  outcomeWashoutDays = 0,
  riskWindowStart = 1,
  startAnchor = "cohort start",
  riskWindowEnd = 365,
  endAnchor = "cohort start",
  covariateSettings = FeatureExtraction::createCovariateSettings(useDemographicsGender =
    TRUE, useDemographicsAge = TRUE, useDemographicsAgeGroup = TRUE, useDemographicsRace
    = TRUE, useDemographicsEthnicity = TRUE, useDemographicsIndexYear = TRUE,
    useDemographicsIndexMonth = TRUE, useDemographicsTimeInCohort = TRUE,
    useDemographicsPriorObservationTime = TRUE, useDemographicsPostObservationTime =
    TRUE, useConditionGroupEraLongTerm = TRUE, useDrugGroupEraOverlapping = TRUE,
    useDrugGroupEraLongTerm = TRUE, useProcedureOccurrenceLongTerm = TRUE,

    useMeasurementLongTerm = TRUE, useObservationLongTerm = TRUE,
    useDeviceExposureLongTerm = TRUE, useVisitConceptCountLongTerm = TRUE,
    useConditionGroupEraShortTerm = TRUE, useDrugGroupEraShortTerm = TRUE,
    useProcedureOccurrenceShortTerm = TRUE, useMeasurementShortTerm = TRUE,
    useObservationShortTerm = TRUE, useDeviceExposureShortTerm = TRUE,
    useVisitConceptCountShortTerm = TRUE, endDays = 0, longTermStartDays = -365,
    shortTermStartDays = -30),
  caseCovariateSettings = createDuringCovariateSettings(useConditionGroupEraDuring =
    TRUE, useDrugGroupEraDuring = TRUE, useProcedureOccurrenceDuring = TRUE,
    useDeviceExposureDuring = TRUE, useMeasurementDuring = TRUE, useObservationDuring =
    TRUE, useVisitConceptCountDuring = TRUE),
  casePreTargetDuration = 365,
  casePostOutcomeDuration = 365,
  extractNonCaseCovariates = TRUE
)
```

Arguments

`targetIds`	A list of cohortIds for the target cohorts
`outcomeIds`	A list of cohortIds for the outcome cohorts

`minPriorObservation`	The minimum time (in days) in the database a patient in the target cohorts must be observed prior to index
`outcomeWashoutDays`	Patients with the outcome within outcomeWashout days prior to index are excluded from the risk factor analysis
`riskWindowStart`	The start of the risk window (in days) relative to the ‘startAnchor‘.
`startAnchor`	The anchor point for the start of the risk window. Can be ‘"cohort start"‘ or ‘"cohort end"‘.
`riskWindowEnd`	The end of the risk window (in days) relative to the ‘endAnchor‘.
`endAnchor`	The anchor point for the end of the risk window. Can be ‘"cohort start"‘ or ‘"cohort end"‘.
`covariateSettings`	An object created using `FeatureExtraction::createCovariateSettings`
`caseCovariateSettings`	An object created using `createDuringCovariateSettings`
`casePreTargetDuration`	The number of days prior to case index we use for FeatureExtraction
`casePostOutcomeDuration`	The number of days prior to case index we use for FeatureExtraction
`extractNonCaseCovariates`	Whether to extract aggregate covariates and counts for patients in the targets and outcomes in addition to the cases

Value

A list with the settings

Examples

```
aggregateSetting <- createAggregateCovariateSettings(
  targetIds = c(1,2),
  outcomeIds = c(3),
  minPriorObservation = 365,
  outcomeWashoutDays = 90,
  riskWindowStart = 1,
  startAnchor = "cohort start",
  riskWindowEnd = 365,
  endAnchor = "cohort start",
  casePreTargetDuration = 365,
  casePostOutcomeDuration = 365
)
```

createCharacterizationSettings

Create the settings for a large scale characterization study

Description

This function creates a list of settings for different characterization studies

Usage

```
createCharacterizationSettings(
  timeToEventSettings = NULL,
  dechallengeRechallengeSettings = NULL,
  aggregateCovariateSettings = NULL
)
```

Arguments

`timeToEventSettings`	A list of timeToEvent settings
`dechallengeRechallengeSettings`	A list of dechallengeRechallenge settings
`aggregateCovariateSettings`	A list of aggregateCovariate settings

Details

Specify one or more timeToEvent, dechallengeRechallenge and aggregateCovariate settings

Value

Returns the connection to the sqlite database

See Also

Other LargeScale: `loadCharacterizationSettings()`, `runCharacterizationAnalyses()`, `saveCharacterizationSet`

Examples

```
# example code

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

cSet <- createCharacterizationSettings(
  dechallengeRechallengeSettings = drSet
)
```

createCharacterizationTables
Create the results tables to store characterization results into a database

Description

This function executes a large set of SQL statements to create tables that can store results

Usage

```
createCharacterizationTables(
  connectionDetails,
  resultSchema,
  targetDialect = "postgresql",
  deleteExistingTables = TRUE,
  createTables = TRUE,
  tablePrefix = "c_",
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

Arguments

`connectionDetails`	The connectionDetails to a database created by using the function `createConnectDetails` in the `DatabaseConnector` package.
`resultSchema`	The name of the database schema that the result tables will be created.
`targetDialect`	The database management system being used
`deleteExistingTables`	If true any existing tables matching the Characterization result tables names will be deleted
`createTables`	If true the Characterization result tables will be created
`tablePrefix`	A string appended to the Characterization result tables
`tempEmulationSchema`	The temp schema used when the database management system is oracle

Details

This function can be used to create (or delete) Characterization result tables

Value

Returns NULL but creates the required tables into the specified database schema.

See Also

Other Database: `createSqliteDatabase()`, `insertResultsToDatabase()`

Examples

```
# create sqlite database
charResultDbCD <- createSqliteDatabase()

# create database results tables
createCharacterizationTables(
   connectionDetails = charResultDbCD,
   resultSchema = 'main'
 )
```

`createDechallengeRechallengeSettings`
Create dechallenge rechallenge study settings

Description

Create dechallenge rechallenge study settings

Usage

```
createDechallengeRechallengeSettings(
  targetIds,
  outcomeIds,
  dechallengeStopInterval = 30,
  dechallengeEvaluationWindow = 30
)
```

Arguments

`targetIds`	A list of cohortIds for the target cohorts
`outcomeIds`	A list of cohortIds for the outcome cohorts
`dechallengeStopInterval`	An integer specifying the how much time to add to the cohort_end when determining whether the event starts during cohort and ends after
`dechallengeEvaluationWindow`	An integer specifying the period of time after the cohort_end when you cannot see an outcome for a dechallenge success

Value

A list with the settings

See Also

Other DechallengeRechallenge: computeDechallengeRechallengeAnalyses(), computeRechallengeFailCaseSeriesA

Examples

```
drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)
```

---

createDuringCovariateSettings

Create during covariate settings

Description

Create during covariate settings

Usage

```
createDuringCovariateSettings(
  useConditionOccurrenceDuring = FALSE,
  useConditionOccurrencePrimaryInpatientDuring = FALSE,
  useConditionEraDuring = FALSE,
  useConditionGroupEraDuring = FALSE,
  useDrugExposureDuring = FALSE,
  useDrugEraDuring = FALSE,
  useDrugGroupEraDuring = FALSE,
  useProcedureOccurrenceDuring = FALSE,
  useDeviceExposureDuring = FALSE,
  useMeasurementDuring = FALSE,
  useObservationDuring = FALSE,
  useVisitCountDuring = FALSE,
  useVisitConceptCountDuring = FALSE,
  includedCovariateConceptIds = c(),
  addDescendantsToInclude = FALSE,
  excludedCovariateConceptIds = c(),
  addDescendantsToExclude = FALSE,
  includedCovariateIds = c()
)
```

Arguments

useConditionOccurrenceDuring	One covariate per condition in the condition_occurrence table starting between cohort start and cohort end. (analysis ID 109)

`useConditionOccurrencePrimaryInpatientDuring`
: One covariate per condition observed as a primary diagnosis in an inpatient setting in the condition_occurrence table starting between cohort start and cohort end. (analysis ID 110)

`useConditionEraDuring`
: One covariate per condition in the condition_era table starting between cohort start and cohort end. (analysis ID 217)

`useConditionGroupEraDuring`
: One covariate per condition era rolled up to groups in the condition_era table starting between cohort start and cohort end. (analysis ID 218)

`useDrugExposureDuring`
: One covariate per drug in the drug_exposure table between cohort start and end. (analysisId 305)

`useDrugEraDuring`
: One covariate per drug in the drug_era table between cohort start and end. (analysis ID 417)

`useDrugGroupEraDuring`
: One covariate per drug rolled up to ATC groups in the drug_era table between cohort start and end. (analysis ID 418)

`useProcedureOccurrenceDuring`
: One covariate per procedure in the procedure_occurrence table between cohort start and end. (analysis ID 505)

`useDeviceExposureDuring`
: One covariate per device in the device exposure table starting between cohort start and end. (analysis ID 605)

`useMeasurementDuring`
: One covariate per measurement in the measurement table between cohort start and end. (analysis ID 713)

`useObservationDuring`
: One covariate per observation in the observation table between cohort start and end. (analysis ID 805)

`useVisitCountDuring`
: The number of visits observed between cohort start and end. (analysis ID 926)

`useVisitConceptCountDuring`
: The number of visits observed between cohort start and end, stratified by visit concept ID. (analysis ID 927)

`includedCovariateConceptIds`
: A list of concept IDs that should be used to construct covariates.

`addDescendantsToInclude`
: Should descendant concept IDs be added to the list of concepts to include?

`excludedCovariateConceptIds`
: A list of concept IDs that should NOT be used to construct covariates.

`addDescendantsToExclude`
: Should descendant concept IDs be added to the list of concepts to exclude?

`includedCovariateIds`
: A list of covariate IDs that should be restricted to.

Details

creates an object specifying how during covariates should be constructed from data in the CDM model.

Value

An object of type `covariateSettings`, to be used in other functions.

See Also

Other CovariateSetting: `getDbDuringCovariateData()`

Examples

```
settings <- createDuringCovariateSettings(
  useConditionOccurrenceDuring = TRUE,
  useConditionOccurrencePrimaryInpatientDuring = FALSE,
  useConditionEraDuring = FALSE,
  useConditionGroupEraDuring = FALSE
)
```

`createSqliteDatabase` *Create an sqlite database connection*

Description

This function creates a connection to an sqlite database

Usage

```
createSqliteDatabase(sqliteLocation = tempdir())
```

Arguments

`sqliteLocation`	The location of the sqlite database

Details

This function creates a sqlite database and connection

Value

Returns the connection detail object to the sqlite database

See Also

Other Database: `createCharacterizationTables()`, `insertResultsToDatabase()`

Examples

```
charResultDbCD <- createSqliteDatabase()
```

`createTimeToEventSettings` *Create time to event study settings*

Description

Create time to event study settings

Usage

```
createTimeToEventSettings(targetIds, outcomeIds)
```

Arguments

`targetIds`	A list of cohortIds for the target cohorts
`outcomeIds`	A list of cohortIds for the outcome cohorts

Value

An list with the time to event settings

See Also

Other TimeToEvent: `computeTimeToEventAnalyses()`

Examples

```
# example code

tteSet <- createTimeToEventSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)
```

exampleOmopConnectionDetails

create a connection detail for an example GI Bleed dataset from Eunomia

Description

This returns an object of class ‘ConnectionDetails‘ that lets you connect via ‘DatabaseConnector::connect()‘ to the example database.

Usage

```
exampleOmopConnectionDetails(exdir = tempdir())
```

Arguments

`exdir`	a directory to unzip the example OMOP database into. Default is tempdir().

Details

Finds the location of the example database in the package and calls ‘DatabaseConnector::createConnectionDetails‘ to create a ‘ConnectionDetails‘ object for connecting to the database.

Value

An object of class ‘ConnectionDetails‘ with the details to connect to the example OHDSI OMOP CDM database

Examples

```
conDet <- exampleOmopConnectionDetails()

connectionHandler <- ResultModelManager::ConnectionHandler$new(conDet)
```

getDbDuringCovariateData

Extracts covariates that occur during a cohort

Description

Extracts covariates that occur during a cohort

Usage

```
getDbDuringCovariateData(
  connection,
  oracleTempSchema = NULL,
  cdmDatabaseSchema,
  cdmVersion = "5",
  cohortTable = "#cohort_person",
  rowIdField = "subject_id",
  aggregated = TRUE,
  cohortIds = c(-1),
  covariateSettings,
  minCharacterizationMean = 0,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  ...
)
```

Arguments

`connection`	The database connection
`oracleTempSchema`	The temp schema if using oracle
`cdmDatabaseSchema`	The schema of the OMOP CDM data
`cdmVersion`	version of the OMOP CDM data
`cohortTable`	the table name that contains the target population cohort
`rowIdField`	string representing the unique identifier in the target population cohort
`aggregated`	whether the covariate should be aggregated
`cohortIds`	cohort id for the target cohort
`covariateSettings`	settings for the covariate cohorts and time periods
`minCharacterizationMean`	the minimum value for a covariate to be extracted
`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
`...`	additional arguments from FeatureExtraction

Details

The user specifies a what during covariates they want and this executes them using FE

Value

A 'FeatureExtraction' covariateData object containing the during covariates based on user settings

See Also

Other CovariateSetting: `createDuringCovariateSettings()`

Examples

```
conDet <- exampleOmopConnectionDetails()
connection <- DatabaseConnector::connect(conDet)

settings <- createDuringCovariateSettings(
  useConditionOccurrenceDuring = TRUE,
  useConditionOccurrencePrimaryInpatientDuring = FALSE,
  useConditionEraDuring = FALSE,
  useConditionGroupEraDuring = FALSE
)

duringData <- getDbDuringCovariateData(
  connection <- connection,
  cdmDatabaseSchema = 'main',
  cohortIds = 1,
  covariateSettings = settings,
  cohortTable = 'cohort'
)
```

---

insertResultsToDatabase

Upload the results into a result database

Description

This function uploads results in csv format into a result database

Usage

```
insertResultsToDatabase(
  connectionDetails,
  schema,
  resultsFolder,
  tablePrefix = "",
  csvTablePrefix = "c_"
)
```

Arguments

`connectionDetails`	The connection details to the result database
`schema`	The schema for the result database
`resultsFolder`	The folder containing the csv results

`tablePrefix`	A prefix to append to the result tables for the characterization results
`csvTablePrefix`	The prefix added to the csv results - default is 'c_'

Details

Calls ResultModelManager uploadResults function to upload the csv files

Value

Returns the connection to the sqlite database

See Also

Other Database: `createCharacterizationTables()`, `createSqliteDatabase()`

Examples

```
# generate results into resultsFolder
conDet <- exampleOmopConnectionDetails()

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

cSet <- createCharacterizationSettings(
  dechallengeRechallengeSettings = drSet
)

runCharacterizationAnalyses(
  connectionDetails = conDet,
  targetDatabaseSchema = 'main',
  targetTable = 'cohort',
  outcomeDatabaseSchema = 'main',
  outcomeTable = 'cohort',
  cdmDatabaseSchema = 'main',
  characterizationSettings = cSet,
  outputDirectory = tempdir()
)

# create sqlite database
charResultDbCD <- createSqliteDatabase()

# create database results tables
createCharacterizationTables(
   connectionDetails = charResultDbCD,
   resultSchema = 'main'
 )

# insert results
insertResultsToDatabase(
 connectionDetails = charResultDbCD,
 schema = 'main',
 resultsFolder = tempdir()
)
```

---

loadCharacterizationSettings

Load the characterization settings previously saved as a json file

---

Description

This function converts the json file back into an R object

Usage

```
loadCharacterizationSettings(fileName)
```

Arguments

`fileName`	The location of the the json settings

Details

Input the directory containing the 'characterizationSettings.json' file and load the settings into R

Value

Returns the json settings as an R object

See Also

Other LargeScale: `createCharacterizationSettings()`, `runCharacterizationAnalyses()`, `saveCharacterizationSettings()`

Examples

```
# example code

setPath <- file.path(tempdir(), 'charSet.json')

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

cSet <- createCharacterizationSettings(
  dechallengeRechallengeSettings = drSet
)
```

```
saveCharacterizationSettings(
  settings = cSet,
  fileName = setPath
)

setting <- loadCharacterizationSettings(setPath)
```

---

runCharacterizationAnalyses

execute a large-scale characterization study

Description

Specify the database connection containing the CDM data, the cohort database schemas/tables, the characterization settings and the directory to save the results to

Usage

```
runCharacterizationAnalyses(
  connectionDetails,
  targetDatabaseSchema,
  targetTable,
  outcomeDatabaseSchema,
  outcomeTable,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cdmDatabaseSchema,
  characterizationSettings,
  outputDirectory,
  executionPath = file.path(outputDirectory, "execution"),
  csvFilePrefix = "c_",
  databaseId = "1",
  showSubjectId = FALSE,
  minCellCount = 0,
  incremental = TRUE,
  threads = 1,
  minCharacterizationMean = 0.01
)
```

Arguments

`connectionDetails`	The connection details to the database containing the OMOP CDM data
`targetDatabaseSchema`	Schema name where your target cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`targetTable`	Name of the target cohort table.
`outcomeDatabaseSchema`	Schema name where your outcome cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`outcomeTable`	Name of the outcome cohort table.
`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
`cdmDatabaseSchema`	The schema with the OMOP CDM data
`characterizationSettings`	The study settings created using `createCharacterizationSettings`
`outputDirectory`	The location to save the final csv files to
`executionPath`	The location where intermediate results are saved to
`csvFilePrefix`	A string to append the csv files in the outputDirectory
`databaseId`	The unique identifier for the cdm database
`showSubjectId`	Whether to include subjectId of failed rechallenge case series or hide
`minCellCount`	The minimum count value that is calculated
`incremental`	If TRUE then skip previously executed analyses that completed
`threads`	The number of threads to use when running aggregate covariates
`minCharacterizationMean`	The minimum mean threshold to extract when running aggregate covariates

Details

The results of the characterization will be saved into an sqlite database inside the specified saveDirectory

Value

Multiple csv files in the outputDirectory.

See Also

Other LargeScale: `createCharacterizationSettings()`, `loadCharacterizationSettings()`, `saveCharacterizationSettings()`

Examples

```
conDet <- exampleOmopConnectionDetails()

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

cSet <- createCharacterizationSettings(
  dechallengeRechallengeSettings = drSet
)

runCharacterizationAnalyses(
  connectionDetails = conDet,
  targetDatabaseSchema = 'main',
  targetTable = 'cohort',
  outcomeDatabaseSchema = 'main',
  outcomeTable = 'cohort',
  cdmDatabaseSchema = 'main',
  characterizationSettings = cSet,
  outputDirectory = tempdir()
)
```

saveCharacterizationSettings

Save the characterization settings as a json

Description

This function converts the settings into a json object and saves it

Usage

```
saveCharacterizationSettings(settings, fileName)
```

Arguments

`settings`	An object of class characterizationSettings created using `createCharacterizationSettings`
`fileName`	The location to save the json settings

Details

Input the characterization settings and output a json file to a file named 'characterizationSettings.json' inside the saveDirectory

Value

Returns the location of the directory containing the json settings

See Also

Other LargeScale: `createCharacterizationSettings()`, `loadCharacterizationSettings()`, `runCharacterizationAnalyses()`

Examples

```
drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)

cSet <- createCharacterizationSettings(
  dechallengeRechallengeSettings = drSet
)

saveCharacterizationSettings(
  settings = cSet,
  fileName = file.path(tempdir(), 'cSet.json')
)
```

`viewCharacterization` *viewCharacterization - Interactively view the characterization results*

Description

This is a shiny app for viewing interactive plots and tables

Usage

```
viewCharacterization(resultFolder, cohortDefinitionSet = NULL)
```

Arguments

`resultFolder`	The location of the csv results
`cohortDefinitionSet`	The cohortDefinitionSet extracted using webAPI

Details

Input is the output of ...

Value

Opens a shiny app for interactively viewing the results

Examples

```
conDet <- exampleOmopConnectionDetails()

drSet <- createDechallengeRechallengeSettings(
  targetIds = c(1,2),
  outcomeIds = 3
)
```

```
cSet <- createCharacterizationSettings(
  dechallengeRechallengeSettings = drSet
)

runCharacterizationAnalyses(
  connectionDetails = conDet,
  targetDatabaseSchema = 'main',
  targetTable = 'cohort',
  outcomeDatabaseSchema = 'main',
  outcomeTable = 'cohort',
  cdmDatabaseSchema = 'main',
  characterizationSettings = cSet,
  outputDirectory = file.path(tempdir(),'view')
)

viewCharacterization(
  resultFolder = file.path(tempdir(),'view')
)
```

Index