Package 'CohortDiagnostics'

December 6, 2022

Type Package

Title Diagnostics for OHDSI Cohorts

Version 3.1.1

Date 2022-07-20

Maintainer Jamie Gilbert <gilbert@ohdsi.org>

Description CohortDiagnostics is an R utility package for the development and evaluation of phenotype algorithms for
OMOP CDM compliant data sets. This package provides a standard, end to end, set of analytics for understanding
patient capture including data generation and result exploration through an R Shiny interface.
Analytics computed include cohort characteristics, record counts, index event misclassification, captured
observation windows and basic incidence proportions for age, gender and calendar year.
Through the identification of errors, CohortDiagnostics enables the comparison of multiple candidate cohort
definitions across one or more data sources, facilitating reproducible research.

Depends DatabaseConnector (>= 5.0.0),
FeatureExtraction (>= 3.2.0),
R (>= 4.1.0)

Imports Andromeda,
ResultModelManager,
checkmate,
clock,
digest,
dplyr (>= 1.0.0),
methods,
ParallelLogger (>= 3.0.0),
readr (>= 2.1.0),
RJSONIO,
rlang,
SqlRender (>= 1.9.0),
stringr,
tidyr (>= 1.2.0),
CohortGenerator (>= 0.5.0)

Suggests CirceR,
DT,
Eunomia,

ggplot2,
htmltools,
knitr,
lubridate,
pool,
plotly,
purrr,
RColorBrewer,
remotes,
rmarkdown,
ROhdsiWebApi (>= 1.2.0),
RSQLite (>= 2.2.1),
scales,
shiny,
shinydashboard,
shinyWidgets,
testthat,
withr,
zip

Remotes ohdsi/Eunomia,
ohdsi/FeatureExtraction,
ohdsi/ResultModelManager,
ohdsi/ROhdsiWebApi,
ohdsi/CirceR,
ohdsi/CohortGenerator

License Apache License

VignetteBuilder knitr

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

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

RoxygenNote 7.2.1

Encoding UTF-8

Language en-US

StagedInstall no

R topics documented:

`checkInputFileEncoding` *Check character encoding of input file*

Description

For its input files, CohortDiagnostics only accepts UTF-8 or ASCII character encoding. This function can be used to check whether a file meets these criteria.

Usage

```
checkInputFileEncoding(fileName)
```

Arguments

`fileName`	The path to the file to check

Value

Throws an error if the input file does not have the correct encoding.

`createDiagnosticsExplorerZip` *Create publishable shiny zip*

Description

A utility designed for creating a published zip of a shiny app with an sqlite database. Designed for sharing projects on servers like data.ohdsi.org.

Takes the shiny code from the R project and adds an sqlite file to a zip archive. Uncompressed cohort diagnostics sqlite databases can become large very quickly.

Usage

```
createDiagnosticsExplorerZip(
  outputZipfile = file.path(getwd(), "DiagnosticsExplorer.zip"),
  sqliteDbPath = "MergedCohortDiagnosticsData.sqlite",
 shinyDirectory = system.file(file.path("shiny", "DiagnosticsExplorer"), package =
    "CohortDiagnostics"),
  overwrite = FALSE
)
```

Arguments

outputZipfile	The output path for the zip file
sqliteDbPath	Merged Cohort Diagnostics sqlitedb created with createMergedResultsFile
shinyDirectory	(optional) Path to the location where the shiny code is stored. By default, this is the package root
overwrite	If the zip file already exists, overwrite it?

---

createMergedResultsFile

Merge Shiny diagnostics files into sqlite database

Description

This function combines diagnostics results from one or more databases into a single file. The result is an sqlite database that can be used as input for the Diagnostics Explorer Shiny app.

It also checks whether the results conform to the results data model specifications.

Usage

```
createMergedResultsFile(
  dataFolder,
  sqliteDbPath = "MergedCohortDiagnosticsData.sqlite",
  overwrite = FALSE,
  tablePrefix = ""
)
```

Arguments

dataFolder	folder where the exported zip files for the diagnostics are stored. Use the executeDiagnostics function to generate these zip files. Zip files containing results from multiple databases may be placed in the same folder.
sqliteDbPath	Output path where sqlite database is placed
overwrite	(Optional) overwrite existing sqlite lite db if it exists.
tablePrefix	(Optional) string to insert before table names (e.g. "cd_") for database table names

createResultsDataModel

Create the results data model tables on a database server.

Description

Create the results data model tables on a database server.

Usage

```
createResultsDataModel(
  connectionDetails = NULL,
  databaseSchema,
  tablePrefix = ""
)
```

Arguments

connectionDetails	DatabaseConnector connectionDetails instance @seealso[DatabaseConnector::createConnecti
databaseSchema	The schema on the postgres server where the tables will be created.
tablePrefix	(Optional) string to insert before table names (e.g. "cd_") for database table names

Details

Only PostgreSQL servers are supported.

executeDiagnostics

Execute cohort diagnostics

Description

Runs the cohort diagnostics on all (or a subset of) the cohorts instantiated using the Assumes the cohorts have already been instantiated. with the CohortGenerator package

Characterization: If runTemporalCohortCharacterization argument is TRUE, then the following default covariateSettings object will be created using `RFeatureExtraction::createTemporalCovariateS` Alternatively, a covariate setting object may be created using the above as an example.

Usage

```
executeDiagnostics(
  cohortDefinitionSet,
  exportFolder,
  databaseId,
  cohortDatabaseSchema,
  databaseName = NULL,
  databaseDescription = NULL,
  connectionDetails = NULL,
```

```
  connection = NULL,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cohortTable = "cohort",
 cohortTableNames = CohortGenerator::getCohortTableNames(cohortTable = cohortTable),
  vocabularyDatabaseSchema = cdmDatabaseSchema,
  cohortIds = NULL,
  cdmVersion = 5,
  runInclusionStatistics = TRUE,
  runIncludedSourceConcepts = TRUE,
  runOrphanConcepts = TRUE,
  runTimeSeries = FALSE,
  runVisitContext = TRUE,
  runBreakdownIndexEvents = TRUE,
  runIncidenceRate = TRUE,
  runCohortRelationship = TRUE,
  runTemporalCohortCharacterization = TRUE,
  temporalCovariateSettings = getDefaultCovariateSettings(),
  minCellCount = 5,
  minCharacterizationMean = 0.01,
  incremental = FALSE,
  incrementalFolder = file.path(exportFolder, "incremental")
)
```

Arguments

cohortDefinitionSet	Data.frame of cohorts must include columns cohortId, cohortName, json, sql
exportFolder	The folder where the output will be exported to. If this folder does not exist it will be created.
databaseId	A short string for identifying the database (e.g. 'Synpuf').
cohortDatabaseSchema	Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
databaseName	The full name of the database. If NULL, defaults to value in cdm_source table
databaseDescription	A short description (several sentences) of the database. If NULL, defaults to value in cdm_source table
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
cdmDatabaseSchema	Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database

and schema name, for example 'cdm_data.dbo'.

`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.

`cohortTable`
: Name of the cohort table.

`cohortTableNames`
: Cohort Table names used by CohortGenerator package

`vocabularyDatabaseSchema`
: Schema name where your OMOP vocabulary data resides. This is commonly the same as cdmDatabaseSchema. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.

`cohortIds`
: Optionally, provide a subset of cohort IDs to restrict the diagnostics to.

`cdmVersion`
: The version of the OMOP CDM. Default 5. (Note: only 5 is supported.)

`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.

`minCellCount`
: The minimum cell count for fields contains person counts or fractions.

`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)

`incremental`
: Create only cohort diagnostics that haven't been created before?

`incrementalFolder`
: If `incremental = TRUE`, specify a folder where records are kept of which cohort diagnostics has been executed.

Details

The `cohortSetReference` argument must be a data frame with at least the following columns.These fields will be exported as is to the cohort table that is part of Cohort Diagnostics results data model. Any additional fields found will be stored as JSON object in the metadata field of the cohort table:

cohortId The cohort Id is the id used to identify a cohort definition. This is required to be unique. It will be used to create file names.

cohortName The full name of the cohort. This will be shown in the Shiny app.

json The JSON cohort definition for the cohort.

sql The SQL of the cohort definition rendered from the cohort json.

Examples

```
## Not run:
# Load cohorts (assumes that they have already been instantiated)
cohortTableNames <- CohortGenerator::getCohortTableNames(cohortTable = "cohort")
cohorts <- CohortGenerator::getCohortDefinitionSet(packageName = "MyGreatPackage")
connectionDetails <- createConnectionDetails(
  dbms = "postgresql",
  server = "ohdsi.com",
  port = 5432,
  user = "me",
  password = "secure"
)

executeDiagnostics(
  cohorts = cohorts,
  exportFolder = "export",
  cohortTableNames = cohortTableNames,
  cohortDatabaseSchema = "results",
  cdmDatabaseSchema = "cdm",
  databaseId = "mySpecialCdm",
  connectionDetails = connectionDetails
)

# Use a custom set of cohorts defined in a data.frame
cohorts <- data.frame(
  cohortId = c(100),
  cohortName = c("Cohort Name"),
  logicDescription = c("My Cohort"),
  sql = c(readLines("path_to.sql")),
  json = c(readLines("path_to.json"))
)
executeDiagnostics(
  cohorts = cohorts,
  exportFolder = "export",
  cohortTable = "cohort",
  cohortDatabaseSchema = "results",
  cdmDatabaseSchema = "cdm",
  databaseId = "mySpecialCdm",
  connectionDetails = connectionDetails
)

## End(Not run)
```

`getCdmDataSourceInformation` *Returns information from CDM source table.*

Description

Returns CDM source name, description, release date, CDM release date, version and vocabulary version, where available.

Usage

```
getCdmDataSourceInformation(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema
)
```

Arguments

`connectionDetails`	An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`connection`	An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
`cdmDatabaseSchema`	Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.

Value

Returns a data frame from CDM Data source.

`getCohortCounts` *Count the cohort(s)*

Description

Computes the subject and entry count per cohort

Usage

```
getCohortCounts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  cohortIds = c()
)
```

Arguments

connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the connect function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
cohortDatabaseSchema	Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
cohortTable	Name of the cohort table.
cohortIds	The cohort Id(s) used to reference the cohort in the cohort table. If left empty, all cohorts in the table will be included.

Value

A tibble with cohort counts

---

getDataMigrator *Get database migrations instance*

Description

Returns ResultModelManager DataMigrationsManager instance.

Usage

```
getDataMigrator(connectionDetails, databaseSchema, tablePrefix = "")
```

Arguments

connectionDetails	DatabaseConnector connection details object
databaseSchema	String schema where database schema lives
tablePrefix	(Optional) Use if a table prefix is used before table names (e.g. "cd_")

Value

Instance of ResultModelManager::DataMigrationManager that has interface for converting existing data models

getDefaultCovariateSettings *Get default covariate settings*

Description

Default covariate settings for cohort diagnostics execution

Usage

```
getDefaultCovariateSettings()
```

getDefaultVocabularyTableNames *Get a list of vocabulary table names*

Description

Get a list of vocabulary table names

Usage

```
getDefaultVocabularyTableNames()
```

Value

Get a list of vocabulary table names in results data model

getResultsDataModelSpecifications *Get specifications for Cohort Diagnostics results data model*

Description

Get specifications for Cohort Diagnostics results data model

Usage

```
getResultsDataModelSpecifications()
```

Value

A tibble data frame object with specifications

launchDiagnosticsExplorer *Launch the Diagnostics Explorer Shiny app*

Description

Launch the Diagnostics Explorer Shiny app

Usage

```
launchDiagnosticsExplorer(
  sqliteDbPath = "MergedCohortDiagnosticsData.sqlite",
  connectionDetails = NULL,
  shinyConfigPath = NULL,
  resultsDatabaseSchema = NULL,
  vocabularyDatabaseSchema = NULL,
  vocabularyDatabaseSchemas = resultsDatabaseSchema,
  tablePrefix = "",
  cohortTableName = "cohort",
  databaseTableName = "database",
  aboutText = NULL,
  runOverNetwork = FALSE,
  port = 80,
  launch.browser = FALSE,
  enableAnnotation = TRUE
)
```

Arguments

sqliteDbPath	Path to merged sqlite file. See createMergedResultsFile to create file.
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package, specifying how to connect to the server where the CohortDiagnostics results have been uploaded using the uploadResults function.
shinyConfigPath	Path to shiny yml configuration file (use instead of sqliteDbPath or connectionDetails object)
resultsDatabaseSchema	The schema on the database server where the CohortDiagnostics results have been uploaded.
vocabularyDatabaseSchema	(Deprecated) Please use vocabularyDatabaseSchemas.
vocabularyDatabaseSchemas	(optional) A list of one or more schemas on the database server where the vocabulary tables are located. The default value is the value of the resultsDatabaseSchema. We can provide a list of vocabulary schema that might represent different versions of the OMOP vocabulary tables. It allows us to compare the impact of vocabulary changes on Diagnostics. Not supported with an sqlite database.

tablePrefix (Optional) string to insert before table names (e.g. "cd_") for database table names

cohortTableName
(Optional) if cohort table name differs from the standard - cohort (ignores prefix if set)

databaseTableName
(Optional) if database table name differs from the standard - database (ignores prefix if set)

aboutText Text (using HTML markup) that will be displayed in an About tab in the Shiny app. If not provided, no About tab will be shown.

runOverNetwork (optional) Do you want the app to run over your network?

port (optional) Only used if runOverNetwork = TRUE.

launch.browser Should the app be launched in your default browser, or in a Shiny window. Note: copying to clipboard will not work in a Shiny window.

enableAnnotation
Enable annotation functionality in shiny app

Details

Launches a Shiny app that allows the user to explore the diagnostics

---

migrateDataModel *Migrate Data model*

Description

Migrate data from current state to next state

It is strongly advised that you have a backup of all data (either sqlite files, a backup database (in the case you are using a postgres backend) or have kept the csv/zip files from your data generation.

Usage

```
migrateDataModel(connectionDetails, databaseSchema, tablePrefix = "")
```

Arguments

connectionDetails
DatabaseConnector connection details object

databaseSchema String schema where database schema lives

tablePrefix (Optional) Use if a table prefix is used before table names (e.g. "cd_")

`runCohortRelationshipDiagnostics` *Given a set of cohorts get relationships between the cohorts.*

Description

Given a set of cohorts, get temporal relationships between the cohort_start_date of the cohorts.

Usage

```
runCohortRelationshipDiagnostics(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema = NULL,
  tempEmulationSchema = NULL,
  cohortTable = "cohort",
  targetCohortIds,
  comparatorCohortIds,
  relationshipDays
)
```

Arguments

`connectionDetails`
: An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`connection`
: An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`cohortDatabaseSchema`
: Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`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.

`cohortTable`
: Name of the cohort table.

`targetCohortIds`
: A vector of one or more Cohort Ids for use as target cohorts.

`comparatorCohortIds`
: A vector of one or more Cohort Ids for use as feature/comparator cohorts.

`relationshipDays`
: A dataframe with two columns startDay and endDay representing periods of time to compute relationship

`runCohortTimeSeriesDiagnostics` *Given a set of instantiated cohorts get time series for the cohorts.*

Description

This function first generates a calendar period table, that has calendar intervals between the `timeSeriesMinDate` and `timeSeriesMaxDate`. Calendar Month, Quarter and year are supported. For each of the calendar interval, time series data are computed. The returned object is a R dataframe that will need to be converted to a time series object to perform time series analysis.

Data Source time series: computes time series at the data source level i.e. observation period table. This output is NOT limited to individuals in the cohort table but is for ALL people in the datasource (i.e. present in observation period table)

Usage

```
runCohortTimeSeriesDiagnostics(
  connectionDetails = NULL,
  connection = NULL,
  tempEmulationSchema = NULL,
  cdmDatabaseSchema,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  runCohortTimeSeries = TRUE,
  runDataSourceTimeSeries = FALSE,
  timeSeriesMinDate = as.Date("1980-01-01"),
  timeSeriesMaxDate = as.Date(Sys.Date()),
  stratifyByGender = TRUE,
  stratifyByAgeGroup = TRUE,
  cohortIds = NULL
)
```

Arguments

`connectionDetails`
: An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`connection`
: An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`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`
: Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.

`cohortDatabaseSchema`
: Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortTable`
: Name of the cohort table.

`runCohortTimeSeries`
: Generate and export the cohort level time series?

`runDataSourceTimeSeries`
: Generate and export the Data source level time series? i.e. using all persons found in observation period table.

`timeSeriesMinDate`
: (optional) Minimum date for time series. Default value January 1st 1980.

`timeSeriesMaxDate`
: (optional) Maximum date for time series. Default value System date.

`stratifyByGender`
: Do you want to stratify by Gender

`stratifyByAgeGroup`
: Do you want to stratify by Age group

`cohortIds`
: A vector of one or more Cohort Ids to compute time distribution for.

`takepackageDependencySnapshot` *Take a snapshot of the R environment*

Description

Take a snapshot of the R environment

Usage

```
takepackageDependencySnapshot()
```

Details

This function records all versions used in the R environment as used by runCohortDiagnostics. This function was borrowed from OhdsiRTools

Value

A data frame listing all the dependencies of the root package and their version numbers, in the order in which they should be installed.

timeExecution *Internal utility function for logging execution of variables*

Description

Internal utility function for logging execution of variables

Usage

```
timeExecution(
  exportFolder,
  taskName,
  cohortIds = NULL,
  parent = NULL,
  start = NA,
  execTime = NA,
  expr = NULL
)
```

uploadResults *Upload results to the database server.*

Description

Requires the results data model tables have been created using the createResultsDataModel function.

Set the POSTGRES_PATH environmental variable to the path to the folder containing the psql executable to enable bulk upload (recommended).

Usage

```
uploadResults(
  connectionDetails,
  schema,
  zipFileName,
  forceOverWriteOfSpecifications = FALSE,
  purgeSiteDataBeforeUploading = TRUE,
  tempFolder = tempdir(),
  tablePrefix = ""
)
```

Arguments

connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package.
schema	The schema on the postgres server where the tables have been created.
zipFileName	The name of the zip file.

`forceOverWriteOfSpecifications`
: If TRUE, specifications of the phenotypes, cohort definitions, and analysis will be overwritten if they already exist on the database. Only use this if these specifications have changed since the last upload.

`purgeSiteDataBeforeUploading`
: If TRUE, before inserting data for a specific databaseId all the data for that site will be dropped. This assumes the input zip file contains the full data for that data site.

`tempFolder`
: A folder on the local file system where the zip files are extracted to. Will be cleaned up when the function is finished. Can be used to specify a temp folder on a drive that has sufficient space if the default system temp space is too limited.

`tablePrefix`
: (Optional) string to insert before table names (e.g. "cd_") for database table names