Package 'CohortDiagnostics'

September 18, 2020

Type Package

Title Diagnostics for OHDSI Studies

Version 2.0.0

Date 2020-09-15

Maintainer Gowtham Rao <gowthamrao@gmail.com>

Description
Diagnostics for studies that use the OMOP Common Data Model and the OHDSI tools.

Depends DatabaseConnector (¿= 3.0.0),
R (¿= 3.5.0)

Imports Andromeda,
checkmate,
digest,
dplyr (¿= 1.0.0),
DT,
FeatureExtraction (¿= 3.1.0),
ggplot2,
ParallelLogger (¿= 2.0.0),
plotly,
readr,
rlang,
RJSONIO,
ROhdsiWebApi (¿= 1.1.2),
SqlRender (¿= 1.6.7),
stringr,
tibble (¿= 3.0.0),
tidyr (¿= 1.0.0)

Suggests DT,
Eunomia,
RSQLite (¿ 2.2.0),
htmltools,
knitr,
plotly,
RColorBrewer,
rmarkdown,
scales,
shiny,
shinydashboard,
stringi,

tidyselect,
VennDiagram,
testthat

Remotes ohdsi/Eunomia,
ohdsi/FeatureExtraction,
ohdsi/ROhdsiWebApi,
ohdsi/DatabaseConnector,
r-dbi/RSQLite

License Apache License

VignetteBuilder knitr

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

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

RoxygenNote 7.1.1

Encoding UTF-8

Language en-US

R topics documented:

`breakDownIndexEvents` *Break down index events*

Description

For the concepts included in the index event definition, count how often they are encountered at the cohort index date.

Usage

```
breakDownIndexEvents(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  baseUrl = NULL,
  webApiCohortId = NULL,
  cohortJson = NULL,
  cohortSql = NULL,
  cohortId = cohortId
)
```

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'.

`oracleTempSchema` Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`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.

`baseUrl` The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Needn't be provided if `cohortJson` and `cohortSql` are provided.

`webApiCohortId` A vector of one Cohort Ids as in the source webApi/Atlas.

`cohortJson` A character string containing the JSON of a cohort definition. Needn't be provided if `baseUrl` and `cohortId` are provided.

`cohortSql` The OHDSI SQL representation of the same cohort definition. Needn't be provided if `baseUrl` and `cohortId` are provided.

`cohortId` The cohort ID used to reference the cohort in the cohort table.

Value

A data frame with concepts, and per concept the count of how often the concept was encountered at the index date.

`compareCohortCharacteristics` *Compare cohort characteristics*

Description

Compare the characteristics of two cohorts, computing the standardized difference of the mean.

Usage

```
compareCohortCharacteristics(characteristics1, characteristics2)
```

Arguments

`characteristics1` Characteristics of the first cohort, as created using the `getCohortCharacteristics` function.

`characteristics2` Characteristics of the second cohort, as created using the `getCohortCharacteristics` function.

Value

A data frame comparing the characteristics of the two cohorts.

`compareCovariateValueResult` *Get cohort covariate comparison (including temporal)*

Description

Get cohort covariate value comparison data for cohorts stored in cohort diagnostics results data model. The output of this function may be used, together with covariate ref to create tables/plots regarding a cohort characteristics comparison diagnostics. Because of the large volume of covariates, this function allows to filter range of covariate_value by providing the minimum and maximum proportion. Its important to note that all covariates are expected to output proportions that are between the values 0.0 to 1.0

Usage

```
compareCovariateValueResult(
  connection = NULL,
  connectionDetails = NULL,
  targetCohortIds,
  comparatorCohortIds,
  databaseIds,
  minProportion = 0.01,
  maxProportion = 1,
  isTemporal = TRUE,
  timeIds = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`targetCohortIds`	A vector of one or more Cohort Ids.
`comparatorCohortIds`	A vector of one or more Cohort Ids.
`databaseIds`	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.
`minProportion`	Do you want to limit the data returned by a lower threshold. Enter a number between 0.00 to 1.00. Be default the value is 0.01.

`maxProportion` Do you want to limit the data returned by a upper threshold. Enter a number between 0.00 to 1.00. Be default the value is 1.

`isTemporal` (optional) Get temporal covariate values?

`timeIds` (optional) Used only if isTemporal = TRUE. Do you want to limit to certain 'time ids'. By default all time ids are returned.

`resultsDatabaseSchema` (optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
covariateValue <- compareCovariateValueResult(targetCohortIds = c(342432,432423),
                                              comparatorCohortIds = c(34243, 342432),
                                              databaseIds = c('eunomia', 'hcup'))

## End(Not run)
```

`computeCohortOverlap` *Compute overlap between two cohorts*

Description

Computes the overlap between a target and a comparator cohort.

Usage

```
computeCohortOverlap(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  targetCohortId,
  comparatorCohortId
)
```

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.
targetCohortId	The cohort ID used to reference the target cohort in the cohort table.
comparatorCohortId	The cohort ID used to reference the comparator cohort in the cohort table.

Value

A data frame with overlap statistics.

---

createCohortTable *Create cohort table(s)*

Description

This function creates an empty cohort table. Optionally, additional empty tables are created to store statistics on the various inclusion criteria.

Usage

```
createCohortTable(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  createInclusionStatsTables = FALSE,
  resultsDatabaseSchema = cohortDatabaseSchema,
```

```
  cohortInclusionTable = paste0(cohortTable, "_inclusion"),
  cohortInclusionResultTable = paste0(cohortTable, "_inclusion_result"),
  cohortInclusionStatsTable = paste0(cohortTable, "_inclusion_stats"),
  cohortSummaryStatsTable = paste0(cohortTable, "_summary_stats")
)
```

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.
createInclusionStatsTables	Create the four additional tables for storing inclusion rule statistics?
resultsDatabaseSchema	Schema name where the statistics tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
cohortInclusionTable	Name of the inclusion table, one of the tables for storing inclusion rule statistics.
cohortInclusionResultTable	Name of the inclusion result table, one of the tables for storing inclusion rule statistics.
cohortInclusionStatsTable	Name of the inclusion stats table, one of the tables for storing inclusion rule statistics.
cohortSummaryStatsTable	Name of the summary stats table, one of the tables for storing inclusion rule statistics.

---

createConceptCountsTable

Create concept counts table

Description

Create a table with counts of how often each concept ID occurs in the CDM.

Usage

```
createConceptCountsTable(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  conceptCountsDatabaseSchema = cdmDatabaseSchema,
  conceptCountsTable = "concept_counts",
  conceptCountsTableIsTemp = FALSE
)
```

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'.
oracleTempSchema	Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.
conceptCountsDatabaseSchema	Schema name where your concept counts table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'. Ignored if conceptCountsTableIsTemp = TRUE.
conceptCountsTable	Name of the concept counts table. This table can be created using the createConceptCountsTable.
conceptCountsTableIsTemp	Is the concept counts table a temp table?

---

createDdl *Create a DDL script for results data model from specification csv.*

Description

Create a DDL script for results data model from specification csv.

Usage

```
createDdl(packageName, packageVersion, modelVersion, specification)
```

Arguments

packageName	The name of the R package whose output model we are documenting.
packageVersion	The version number of cohort diagnostics
modelVersion	The version of the results data model
specification	The location of the csv file with the high-level results table specification.

`createDdlPkConstraints` *Create DDL with primary key*

Description

Create DDL with primary key

Usage

```
createDdlPkConstraints(
  packageName,
  packageVersion,
  modelVersion,
  specification
)
```

Arguments

packageName	The name of the R package whose output model we are documenting.
packageVersion	The version number of cohort diagnostics
modelVersion	The version of the results data model
specification	The location of the csv file with the high-level results table specification.

`dropDdl` *Create DDL that drops all results table*

Description

Create DDL that drops all results table

Usage

```
dropDdl(packageName, packageVersion, modelVersion, specification)
```

Arguments

packageName	The name of the R package whose output model we are documenting.
packageVersion	The version number of cohort diagnostics
modelVersion	The version of the results data model
specification	The location of the csv file with the high-level results table specification.

`extractConceptSetsJsonFromCohortJson` *Extract concept set json from cohort json*

Description

Extracts json that corresponds to the conceptset definition in a cohort json definition

Usage

```
extractConceptSetsJsonFromCohortJson(cohortJson)
```

Arguments

`cohortJson`	Complete JSON specification of cohort definition. The standard form is generated by `WebApi`

Value

The function will return a tibble data frame object with one row per conceptSet id in cohort definition.

Examples

```
## Not run:
conceptSetsJson <- extractConceptSetsJsonFromCohortJson(cohortJson = json)

## End(Not run)
```

`extractConceptSetsSqlFromCohortSql` *Extract concept set sql from cohort generation SQL*

Description

Extracts SQL that corresponds to the conceptset (codeset) part from cohort generation SQL used to instantiated conceptSets during cohort construction.

Usage

```
extractConceptSetsSqlFromCohortSql(cohortSql)
```

Arguments

`cohortSql`	Complete SQL specification of cohort definition in OHDSI SQL dialect. May contain parameters designed to be replaced by `SqlRender`. The standard form SQL is generated using `circe-be` by `WebApi` and `Atlas`

Value

The function will return a tibble data frame object with one row per concept id and concept set combination in cohort definition.

Examples

```
## Not run:
conceptSetSql <- extractConceptSetsSqlFromCohortSql(cohortSql = sql)

## End(Not run)
```

---

findCohortIncludedSourceConcepts

Check source codes used in a cohort definition

Description

This function first extracts all concept sets used in a cohort definition. Then, for each concept set the concept found in the CDM database the contributing source codes are identified.

Usage

```
findCohortIncludedSourceConcepts(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  baseUrl = NULL,
  webApiCohortId = NULL,
  cohortJson = NULL,
  cohortSql = NULL,
  byMonth = FALSE,
  useSourceValues = FALSE
)
```

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'.

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`baseUrl`
: The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Needn't be provided if `cohortJson` and `cohortSql` are provided.

`webApiCohortId`
: A vector of one Cohort Ids as in the source webApi/Atlas.

`cohortJson`
: A character string containing the JSON of a cohort definition. Needn't be provided if `baseUrl` and `cohortId` are provided.

`cohortSql`
: The OHDSI SQL representation of the same cohort definition. Needn't be provided if `baseUrl` and `cohortId` are provided.

`byMonth`
: Compute counts by month? If FALSE, only overall counts are computed.

`useSourceValues`
: Use the source_value fields to find the codes used in the data? If not, this analysis will rely entirely on the source_concept_id fields instead. Note that, depending on the source data and ETL, it might be possible for the source_value fields to contain patient-identifiable information by accident.

Value

A data frame with source codes, with counts per domain how often the code was encountered in the CDM.

`findCohortOrphanConcepts` *Find orphan concepts for all concept sets in a cohort*

Description

Searches for concepts that should belong to the concept sets in a cohort definition but don't, for example because of missing source-to-standard concept maps, or erroneous hierarchical relationships.

Usage

```
findCohortOrphanConcepts(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  baseUrl = NULL,
  webApiCohortId = NULL,
  cohortJson = NULL,
  conceptCountsDatabaseSchema = cdmDatabaseSchema,
  conceptCountsTable = "concept_counts",
  conceptCountsTableIsTemp = FALSE
)
```

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'.

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`baseUrl`
: The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Needn't be provided if `cohortJson` is provided.

`webApiCohortId`
: The ID of the cohort in the WebAPI instance. Needn't be provided if `cohortJson` is provided.

`cohortJson`
: A character string containing the JSON of a cohort definition. Needn't be provided if `baseUrl` and `webApiCohortId` are provided.

`conceptCountsDatabaseSchema`
: Schema name where your concept counts table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'. Ignored if `conceptCountsTableIsTemp = TRUE`.

`conceptCountsTable`
: Name of the concept counts table. This table can be created using the `createConceptCountsTable`.

`conceptCountsTableIsTemp`
: Is the concept counts table a temp table?

Details

Logically, this function performs the following steps for each concept set expression in the cohort definition:

- Given the concept set expression, find all included concepts.
- Find all names of the input concepts, including synonyms, and the names of source concepts that map to them.
- Search for concepts (standard and source) that contain any of those names as substring.
- Filter those concepts to those that are not in the original set of concepts (i.e. orphans).
- Restrict the set of orphan concepts to those that appear in the CDM database and across network concept prevalence (as either source concept or standard concept).

Value

A data frame with orphan concepts, with counts how often the code was encountered in the CDM.

`findOrphanConcepts` *Find (source) concepts that do not roll up to their ancestor(s)*

Description

Searches for concepts that should belong to the set of concepts but don't, for example because of missing source-to-standard concept maps, or erroneous hierarchical relationships.

Usage

```
findOrphanConcepts(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  conceptIds,
  conceptCountsDatabaseSchema = cdmDatabaseSchema,
  conceptCountsTable = "concept_counts",
  conceptCountsTableIsTemp = FALSE
)
```

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'.

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`conceptIds`
: A vector of concept IDs for which we want to find orphans.

`conceptCountsDatabaseSchema`
: Schema name where your concept counts table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'. Ignored if `conceptCountsTableIsTemp = TRUE`.

`conceptCountsTable`
: Name of the concept counts table. This table can be created using the `createConceptCountsTable`.

`conceptCountsTableIsTemp`
: Is the concept counts table a temp table?

Details

Logically, this function performs the following steps for the input set of concept IDs:

- Find all names of the input concepts, including synonyms, and the names of source concepts that map to them.
- Search for concepts (standard and source) that contain any of those names as substring.
- Filter those concepts to those that are not in the original set of concepts (i.e. orphans).
- Restrict the set of orphan concepts to those that appear in the CDM database and across network concept prevalence (as either source concept or standard concept).

Value

A data frame with orphan concepts, with counts how often the code was encountered in the CDM.

`getCohortCharacteristics` *Create characterization of a cohort*

Description

Computes features using all drugs, conditions, procedures, etc. observed on or prior to the cohort index date.

Usage

```
getCohortCharacteristics(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  cohortIds,
  cdmVersion = 5,
  covariateSettings
)
```

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'.

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`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`
: A vector of cohortIds (1 or more) used to reference the cohort in the cohort table.

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

`covariateSettings`
: Either an object of type `covariateSettings` as created using one of the createCovariate functions in the FeatureExtraction package, or a list of such objects.

Value

A list object with tibbles returned from Feature Extraction

`getCohortCountResult` *Get cohort counts*

Description

Get cohort counts

Usage

```
getCohortCountResult(
  connection = NULL,
  connectionDetails = NULL,
  databaseIds = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`
: (optional) 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.

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

`databaseIds` A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.

`resultsDatabaseSchema`
(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
cohortCounts <- getCohortCountResult(databaseIds = c('eunomia', 'hcup'))

## End(Not run)
```

`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

---

getCohortOverlapResult *Get cohort overlap*

Description

Get cohort overlap data

Usage

```
getCohortOverlapResult(
  connection = NULL,
  connectionDetails = NULL,
  targetCohortIds,
  comparatorCohortIds,
  databaseIds,
  resultsDatabaseSchema = NULL
)
```

Arguments

connection	(optional) 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.

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

`targetCohortIds`
: A vector of one or more Cohort Ids.

`comparatorCohortIds`
: A vector of one or more Cohort Ids.

`databaseIds`
: A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.

`resultsDatabaseSchema`
: (optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
cohortOverlap <- getCohortOverlapResult(targetCohortIds = 342432,
                                        comparatorCohortIds = c(432423,34234),
                                        databaseIds = c('eunomia', 'hcup'))

## End(Not run)
```

`getCohortReference` *Get cohort information*

Description

Get cohort information

Usage

```
getCohortReference(
  connection = NULL,
  connectionDetails = NULL,
  cohortIds = NULL,
  resultsDatabaseSchema = NULL,
  getJson = FALSE,
  getSql = FALSE
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`cohortIds`	A vector of one or more Cohort Ids.
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.
`getJson`	Do you want the JSON expression of cohort?
`getSql`	Do you want the Sql expression of cohort?

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
cohortReference <- getCohortReference()

## End(Not run)
```

---

getCohortsJsonAndSql *Get cohorts JSON and parameterized OHDSI SQL*

Description

This function may be used to collect a cohorts JSON and OHDSI SQL. Based on whether a baseUrl is available, the function will collect the specifications from either from WebApi or a Package.

Usage

```
getCohortsJsonAndSql(
  packageName = NULL,
  cohortToCreateFile = "settings/CohortsToCreate.csv",
  baseUrl = NULL,
  cohortSetReference = NULL,
  cohortIds = NULL
)
```

Arguments

packageName	The name of the package containing the cohort definitions. Can be left NULL if baseUrl and cohortSetReference have been specified.
cohortToCreateFile	The location of the cohortToCreate file within the package. Is ignored if baseUrl and cohortSetReference have been specified. The cohortToCreateFile must be .csv file that is expected to be read into a dataframe object identical to requirements for cohortSetReference argument.
baseUrl	The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Can be left NULL if packageName and cohortToCreateFile have been specified.
cohortSetReference	A data frame with four columns, as described in the details. Can be left NULL if packageName and cohortToCreateFile have been specified.
cohortIds	Optionally, provide a subset of cohort IDs to restrict the diagnostics to.

Details

Currently two ways of executing this function are supported, either (1) [Package Mode] embedded in a study package, assuming the cohort definitions are stored in that package using the `ROhdsiWebApi::insertCohortDefinitionSetInPackage`, or (2) [WebApi Mode] By using a WebApi interface to retrieve the cohort definitions.

When using this function in Package Mode: Use the `packageName` and `cohortToCreateFile` to specify the name of the study package, and the name of the cohortToCreate file within that package, respectively

When using this function in WebApi Mode: use the `baseUrl` and `cohortSetReference` to specify how to connect to the WebApi, and which cohorts to fetch, respectively.

Note: if the parameters for both Package Mode and WebApi Mode are provided, then Package mode is preferred.

The `cohortSetReference` argument must be a data frame with the following columns:

referentConceptId A standard omop concept id that serves as the referant phenotype definition for the cohort Id.

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. It is recommended to be (referrentConceptId * 1000) + a number between 3 to 999

webApiCohortId Cohort Id in the webApi/atlas instance. It is a required field to run Cohort Diagnostics in WebApi mode. It is discarded in package mode.

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

logicDescription A human understandable brief description of the cohort definition. This logic does not have to a fully specified description of the cohort definition, but should provide enough context to help user understand the meaning of the cohort definition

Value

The function will return a R list object with cohort information including specifications such as JSON and SQL.

Examples

```
## Not run:
cohorts <- getCohortsJsonAndSql(packageName = 'cohortDiagnostics',
                                baseUrl = "http://server.org:80/WebAPI")

## End(Not run)
```

`getConceptReference` *Get concept information*

Description

Get concept information

Usage

```
getConceptReference(
  connection = NULL,
  connectionDetails = NULL,
  conceptIds = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

connection	(optional) 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.
connectionDetails	(optional) An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
conceptIds	(optional) A vector of integers corresponding to conceptids of interest.
resultsDatabaseSchema	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when connectionDetails or connect is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either connectionDetails or connect parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both connectionDetails and connect are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
conceptReference <- getConceptReference()

## End(Not run)
```

`getConceptSetDataDiagnostics` *Get concept set data diagnostics*

Description

Get concept set data diagnostics data

Usage

```
getConceptSetDataDiagnostics(
  connection = NULL,
  connectionDetails = NULL,
  cohortIds = NULL,
  databaseIds = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`cohortIds`	A vector of one or more Cohort Ids.
`databaseIds`	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data

frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
conceptSetDataDiagnostics <- getConceptSetDataDiagnostics()

## End(Not run)
```

---

getCovariateReference *Get cohort covariate reference (including temporal)*

Description

Get cohort covariate reference (including temporal).

Usage

```
getCovariateReference(
  connection = NULL,
  connectionDetails = NULL,
  covariateIds = NULL,
  isTemporal = TRUE,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`covariateIds`	(optional) A vector of covariateIds to subset the results
`isTemporal`	Get temporal covariate references?
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
covariateReference <- getCovariateReference(isTemporal = FALSE)

## End(Not run)
```

`getCovariateValueResult` *Get cohort covariate (including temporal)*

Description

Get cohort covariate value data from data stored in cohort diagnostics results data model. The output of this function may be used, together with covariate ref (temporal covariate ref and time ref if temporal) for cohorts characteristization diagnostics. Because of the large volume of covariates, this function allows to filter range of covariate_value by providing the minimum and maximum proportion. Its important to note that all covariates are expected to output proportions that are between the values 0.0 to 1.0

Usage

```
getCovariateValueResult(
  connection = NULL,
  connectionDetails = NULL,
  cohortIds,
  databaseIds,
  minProportion = 0.01,
```

```
  maxProportion = 1,
  isTemporal = TRUE,
  timeIds = c(1, 2, 3, 4, 5),
  resultsDatabaseSchema = NULL
)
```

Arguments

connection	(optional) 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.
connectionDetails	(optional) An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
cohortIds	A vector of one or more Cohort Ids.
databaseIds	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.
minProportion	Do you want to limit the data returned by a lower threshold. Enter a number between 0.00 to 1.00. Be default the value is 0.01.
maxProportion	Do you want to limit the data returned by a upper threshold. Enter a number between 0.00 to 1.00. Be default the value is 1.
isTemporal	(optional) Get temporal covariate values?
timeIds	(optional) Will only be used if isTemporal = TRUE. Do you want to limit to certain 'time ids'. By default timeId = c(1,2,3,4,5) are returned. These correspond to -365 to -31, -30 to -1, 0 to 0, 1 to 30, 31 to 365. If any of timeId value = 0, all timeIds are returned. If any of timeId value = -1, will return all timeIds ¿ 5 (for time series analysis)
resultsDatabaseSchema	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when connectionDetails or connect is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either connectionDetails or connect parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both connectionDetails and connect are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
covariateValue <- getCovariateValueResult(cohortIds = c(342432,432423),
                                          databaseIds = c('eunomia', 'hcup'))

## End(Not run)
```

`getDatabaseReference` *Get database information*

Description

Get database information

Usage

```
getDatabaseReference(
  connection = NULL,
  connectionDetails = NULL,
  databaseIds = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`databaseIds`	(optional) A vector of character string to identify database.
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
databaseReference <- getDatabaseReference()

## End(Not run)
```

---

getIncidenceRate *Compute incidence rate for a cohort*

Description

Returns yearly incidence rate stratified by age and gender

Usage

```
getIncidenceRate(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable,
  cdmDatabaseSchema,
  cdmVersion = 5,
  oracleTempSchema = oracleTempSchema,
  firstOccurrenceOnly = TRUE,
  washoutPeriod = 365,
  cohortId
)
```

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.

`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'.

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

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`firstOccurrenceOnly`
: Use only the first occurrence of the cohort per person?

`washoutPeriod`
: The minimum amount of observation time required before the occurrence of a cohort entry. This is also used to eliminate immortal time from the denominator.

`cohortId`
: The cohort Id used to reference the cohort in the cohort table.

Value

Returns a data frame of cohort count, person year count, and incidence rate per 1000 persons years with the following stratifications: 1) no stratification, 2) gender stratification, 3) age (10-year) stratification, 4) calendar year and age (10-year) stratification, 5) calendar year and gender stratification, 6) calendar year, age (10-year), and gender stratification with option to save dataframes.

`getIncidenceRateResult` *Get incidence rate results*

Description

Get incidence rate results.

Usage

```
getIncidenceRateResult(
  connection = NULL,
  connectionDetails = NULL,
  cohortIds,
  databaseIds,
  stratifyByGender = c(TRUE, FALSE),
  stratifyByAgeGroup = c(TRUE, FALSE),
  stratifyByCalendarYear = c(TRUE, FALSE),
  minPersonYears = 1000,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`cohortIds`	A vector of one or more Cohort Ids.
`databaseIds`	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.
`stratifyByGender`	(optional) Do you want to stratify by gender.
`stratifyByAgeGroup`	(optional) Do you want to stratify by age group.
`stratifyByCalendarYear`	(optional) Do you want to stratify by calendar year.
`minPersonYears`	(optional) Default value = 1000. Minimum person years needed to create plot.
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
incidenceRate <- getIncidenceRateResult(cohortIds = 343242,
                                        databaseIds = c('eunomia', 'hcup'))

## End(Not run)
```

`getInclusionStatistics` *Get statistics on cohort inclusion criteria*

Description

Get statistics on cohort inclusion criteria

Usage

```
getInclusionStatistics(
  connectionDetails = NULL,
  connection = NULL,
  resultsDatabaseSchema,
  cohortId,
  simplify = TRUE,
  cohortTable = "cohort",
  cohortInclusionTable = paste0(cohortTable, "_inclusion"),
  cohortInclusionResultTable = paste0(cohortTable, "_inclusion_result"),
  cohortInclusionStatsTable = paste0(cohortTable, "_inclusion_stats"),
  cohortSummaryStatsTable = paste0(cohortTable, "_summary_stats")
)
```

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.

`resultsDatabaseSchema` Schema name where the statistics tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortId` The cohort definition ID used to reference the cohort in the cohort table.

`simplify` Simply output the attrition table?

`cohortTable` Name of the cohort table. Used only to conveniently derive names of the four rule statistics tables.

`cohortInclusionTable` Name of the inclusion table, one of the tables for storing inclusion rule statistics.

`cohortInclusionResultTable` Name of the inclusion result table, one of the tables for storing inclusion rule statistics.

`cohortInclusionStatsTable` Name of the inclusion stats table, one of the tables for storing inclusion rule statistics.

`cohortSummaryStatsTable` Name of the summary stats table, one of the tables for storing inclusion rule statistics.

Value

If 'simplify = TRUE', this function returns a single data frame. Else a list of data frames is returned.

`getInclusionStatisticsFromFiles` *Get inclusion criteria statistics from files*

Description

Gets inclusion criteria statistics from files, as stored when using the `ROhdsiWebApi::insertCohortDefinitionSe` function with `generateStats = TRUE`.

Usage

```
getInclusionStatisticsFromFiles(
  cohortId,
  folder,
  cohortInclusionFile = file.path(folder, "cohortInclusion.csv"),
  cohortInclusionResultFile = file.path(folder, "cohortIncResult.csv"),
  cohortInclusionStatsFile = file.path(folder, "cohortIncStats.csv"),
  cohortSummaryStatsFile = file.path(folder, "cohortSummaryStats.csv"),
  simplify = TRUE
)
```

Arguments

`cohortId`	The cohort definition ID used to reference the cohort in the cohort table.
`folder`	The path to the folder where the inclusion statistics are stored.
`cohortInclusionFile`	Name of the inclusion table, one of the tables for storing inclusion rule statistics.
`cohortInclusionResultFile`	Name of the inclusion result table, one of the tables for storing inclusion rule statistics.
`cohortInclusionStatsFile`	Name of the inclusion stats table, one of the tables for storing inclusion rule statistics.
`cohortSummaryStatsFile`	Name of the summary stats table, one of the tables for storing inclusion rule statistics.
`simplify`	Simply output the attrition table?

Value

If 'simplify = TRUE', this function returns a single data frame. Else a list of data frames is returned.

`getRecordCountOfInstantiatedCohorts`

Get record counts for a set of cohort

Description

This function get record count for a set of cohorts in the cohort table.

Usage

```
getRecordCountOfInstantiatedCohorts(
  connection = NULL,
  connectionDetails = NULL,
  cohortDatabaseSchema,
  cohortTable,
  cohortIds
)
```

Arguments

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

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
: A vecotr of cohortIds to get counts for.

Value

A tibble data frame object

---

getResultsDataModelSpecification *Get specification for Cohort Diagnostics results data model*

Description

Get specification for Cohort Diagnostics results data model

Usage

```
getResultsDataModelSpecification()
```

Value

A tibble data frame object with specifications

Examples

```
## Not run:
resultsDataModelSpecification <- getResultsDataModelSpecification()

## End(Not run)
```

---

getTimeDistributionResult *Get time distribution results*

Description

Get time distribution results

Usage

```
getTimeDistributionResult(
  connection = NULL,
  connectionDetails = NULL,
  cohortIds,
  databaseIds,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`cohortIds`	A vector of one or more Cohort Ids.
`databaseIds`	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
timeDistribution <- getTimeDistributionResult(cohortIds = 343242,
                                              databaseIds = 'eunomia')

## End(Not run)
```

`getTimeDistributions` *Get time distributions for a set of cohorts*

Description

Computes the distribution of the observation time before and after index, and time within a cohort.

Usage

```
getTimeDistributions(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  cohortIds,
  cdmVersion = 5
)
```

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'.
`oracleTempSchema`	Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.
`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`	A vector of cohortIds (1 or more) used to reference the cohort in the cohort table.
`cdmVersion`	The version of the OMOP CDM. Default 5. (Note: only 5 is supported.)

Value

A list object with tibbles returned from Feature Extraction

`getTimeReference` *Get time reference for temporal covariates*

Description

Get time reference for temporal covariates

Usage

```
getTimeReference(
  connection = NULL,
  connectionDetails = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connection`	(optional) 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.
`connectionDetails`	(optional) An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.
`resultsDatabaseSchema`	(optional) The databaseSchema where the results data model of cohort diagnostics is stored. This is only required when `connectionDetails` or `connect` is provided.

Details

Note: The output of this function may be used to create plots or tables. This function relies on data available in Cohort Diagnostics results data model. The function will use one of the two methods to connect to the results data model, 1) database mode, and 2) in-memory mode.

Database mode: In this mode, R will look for the results data model in a remote relational dbms. The database system should be one of the databases supported by DatabaseConnector package. If either `connectionDetails` or `connect` parameters are populated with an argument during a function call, the connection gets set to database mode.

In-memory mode: If both `connectionDetails` and `connect` are parameters have a NULL argument then query will be in-memory mode. R will now expect and check for a data frame object available in R's memory, as required for the function. The object should be an object specificed in Cohort Diagnostics results data model.

Objects in R's memory are expected to follow camelCase naming conventions (see HADES), while objects in a relational database system are expected to follow snake-case naming conventions.

Value

The function will return a tibble data frame object.

Examples

```
## Not run:
timeReference <- getTimeReference()

## End(Not run)
```

`getUniqueConceptIds` *Get all unique concept id's referenced in the cohort diagnostics*

Description

Get all unique concept id's referenced in the cohort diagnostics

Usage

```
getUniqueConceptIds(exportFolder)
```

Arguments

`exportFolder`	The folder where the output is exported by Cohort Diagnostics. If this folder does not exist, or does not have the searched file the function will return an error.

Value

Returns a vector unique conceptId's from various objects in the export folder.

`guessCsvFileSpecification`
Guesses data model specification from multiple csv files.

Description

Guesses data model specification from multiple csv files.

Usage

```
guessCsvFileSpecification(pathToCsvFile)
```

Arguments

`pathToCsvFile`	file system path to csv file

Value

A tibble data frame object with specifications

Examples

```
## Not run:
csvFileSpecification <- guessCsvFileSpecification(path)

## End(Not run)
```

`instantiateCohort` *Instantiate a cohort*

Description

This function instantiates the cohort in the cohort table. Optionally, the inclusion rule statistics are computed and stored in the inclusion rule statistics tables described in `createCohortTable`).

Usage

```
instantiateCohort(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  baseUrl = NULL,
  cohortJson = NULL,
  cohortSql = NULL,
  cohortId = NULL,
  generateInclusionStats = FALSE,
  resultsDatabaseSchema = cohortDatabaseSchema,
  cohortInclusionTable = paste0(cohortTable, "_inclusion"),
  cohortInclusionResultTable = paste0(cohortTable, "_inclusion_result"),
  cohortInclusionStatsTable = paste0(cohortTable, "_inclusion_stats"),
  cohortSummaryStatsTable = paste0(cohortTable, "_summary_stats")
)
```

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'.

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

`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.

`baseUrl`
: The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Needn't be provided if `cohortJson` and `cohortSql` are provided.

`cohortJson`
: A character string containing the JSON of a cohort definition. Needn't be provided if `baseUrl` and `cohortId` are provided.

`cohortSql`
: The OHDSI SQL representation of the same cohort definition. Needn't be provided if `baseUrl` and `cohortId` are provided.

`cohortId`
: The cohort ID used to reference the cohort in the cohort table.

`generateInclusionStats`
: Compute and store inclusion rule statistics?

`resultsDatabaseSchema`
: Schema name where the statistics tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortInclusionTable`
: Name of the inclusion table, one of the tables for storing inclusion rule statistics.

`cohortInclusionResultTable`
: Name of the inclusion result table, one of the tables for storing inclusion rule statistics.

`cohortInclusionStatsTable`
: Name of the inclusion stats table, one of the tables for storing inclusion rule statistics.

`cohortSummaryStatsTable`
: Name of the summary stats table, one of the tables for storing inclusion rule statistics.

`instantiateCohortSet` *Instantiate a set of cohort*

Description

This function instantiates a set of cohort in the cohort table, using definitions that are fetched from a WebApi interface. Optionally, the inclusion rule statistics are computed and stored in the `inclusionStatisticsFolder`.

Usage

```
instantiateCohortSet(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
```

```
  oracleTempSchema = NULL,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  cohortIds = NULL,
  packageName = NULL,
  cohortToCreateFile = "settings/CohortsToCreate.csv",
  baseUrl = NULL,
  cohortSetReference = NULL,
  generateInclusionStats = FALSE,
  inclusionStatisticsFolder = NULL,
  createCohortTable = FALSE,
  incremental = FALSE,
  incrementalFolder = 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.

`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'.

`oracleTempSchema`
: Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.

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

`packageName`
: The name of the package containing the cohort definitions. Can be left NULL if `baseUrl` and `cohortSetReference` have been specified.

`cohortToCreateFile`
: The location of the cohortToCreate file within the package. Is ignored if `baseUrl` and `cohortSetReference` have been specified. The cohortToCreateFile must be .csv file that is expected to be read into a dataframe object identical to requirements for `cohortSetReference` argument.

`baseUrl`
: The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Can be left NULL if `packageName` and `cohortToCreateFile` have been specified.

`cohortSetReference`
: A data frame with four columns, as described in the details. Can be left NULL if `packageName` and `cohortToCreateFile` have been specified.

`generateInclusionStats`
: Compute and store inclusion rule statistics?

`inclusionStatisticsFolder`
: The folder where the inclusion rule statistics are stored. Can be left NULL if `generateInclusionStats = FALSE`.

`createCohortTable`
: Create the cohort table? If `incremental = TRUE` and the table already exists this will be skipped.

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

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

Details

Currently two ways of executing this function are supported, either (1) [Package Mode] embedded in a study package, assuming the cohort definitions are stored in that package using the `ROhdsiWebApi::insertCohortDefinitionSetInPackage`, or (2) [WebApi Mode] By using a WebApi interface to retrieve the cohort definitions.

When using this function in Package Mode: Use the `packageName` and `cohortToCreateFile` to specify the name of the study package, and the name of the cohortToCreate file within that package, respectively

When using this function in WebApi Mode: use the `baseUrl` and `cohortSetReference` to specify how to connect to the WebApi, and which cohorts to fetch, respectively.

Note: if the parameters for both Package Mode and WebApi Mode are provided, then Package mode is preferred.

The `cohortSetReference` argument must be a data frame with the following columns:

referentConceptId A standard omop concept id that serves as the referant phenotype definition for the cohort Id.

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. It is recommended to be (referrentConceptId * 1000) + a number between 3 to 999

webApiCohortId Cohort Id in the webApi/atlas instance. It is a required field to run Cohort Diagnostics in WebApi mode. It is discarded in package mode.

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

logicDescription A human understandable brief description of the cohort definition. This logic does not have to a fully specified description of the cohort definition, but should provide enough context to help user understand the meaning of the cohort definition

Value

A data frame with cohort counts

launchCohortExplorer *Launch the CohortExplorer Shiny app*

Description

Launch the CohortExplorer Shiny app

Usage

```
launchCohortExplorer(
  connectionDetails,
  cdmDatabaseSchema,
  cohortDatabaseSchema,
  cohortTable,
  cohortId,
  sampleSize = 100,
  subjectIds = NULL
)
```

Arguments

connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package.
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.
cohortId	The ID of the cohort.
sampleSize	Number of subjects to sample from the cohort. Ignored if subjectIds is specified.
subjectIds	A vector of subject IDs to view.

Details

Launches a Shiny app that allows the user to explore a cohort of interest.

`launchDiagnosticsExplorer` *Launch the Diagnostics Explorer Shiny app*

Description

Launch the Diagnostics Explorer Shiny app

Usage

```
launchDiagnosticsExplorer(dataFolder, launch.browser = FALSE)
```

Arguments

`dataFolder`	A folder where the exported zip files for the diagnostics are stored. Use the `runCohortDiagnostics` function to generate these zip files. Zip files containing results from multiple databases can be placed in the same folder.
`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.

Details

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

---

`plotCohortComparisonStandardizedDifference` *Get Plotly object with cohort comparison plot.*

Description

Get Plotly object with cohort comparison plot.

Usage

```
plotCohortComparisonStandardizedDifference(
  data,
  targetCohortIds = NULL,
  comparatorCohortIds = NULL,
  cohortReference = NULL,
  covariateReference = NULL,
  concept = NULL,
  absoluteStandardizedDifferenceLowerThreshold = 0.001,
  absoluteStandardizedDifferenceUpperThreshold = 1,
  databaseIds = NULL
)
```

Arguments

data	A tibble data frame object that is the output of compareCovariateValueResult function.
targetCohortIds	(optional) A vector of one or more Cohort Ids.
comparatorCohortIds	(optional) A vector of one or more Cohort Ids.
cohortReference	(optional) A tibble data frame object returned from getCohortReference function.
covariateReference	(optional) A tibble data frame object returned from getCovariateReference function.
concept	(optional) A tibble data frame object returned from getConceptReference function.
absoluteStandardizedDifferenceLowerThreshold	(optional) Do you want to keep a lower threshold of absolute standardized difference for plotting
absoluteStandardizedDifferenceUpperThreshold	(optional) Do you want to keep a lower threshold of absolute standardized difference for plotting
databaseIds	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.

Value

A Plotly object.

Examples

```
## Not run:
plotCohortCompare <- plotCohortComparisonStandardizedDifference(data = data)

## End(Not run)
```

---

plotCohortOverlapVennDiagram *Get Vendiagram object with cohort Overlap plot.*

Description

Get Vendiagram object with cohort Overlap plot.

Usage

```
plotCohortOverlapVennDiagram(
  data,
  targetCohortIds,
  comparatorCohortIds,
  databaseIds
)
```

Arguments

data	A tibble data frame object that is the output of getCohortOverlapResult function.
targetCohortIds	(optional) A vector of one or more Cohort Ids.
comparatorCohortIds	(optional) A vector of one or more Cohort Ids.
databaseIds	A vector one or more databaseIds to retrieve the results for. This is a character field values from the 'databaseId' field of the 'database' table of the results data model.

Value

A Vendiagram object.

Examples

```
## Not run:
plotCohortOverlapVennDiagram <- plotCohortOverlapVennDiagram(data = data)

## End(Not run)
```

---

plotIncidenceRate *Get ggplot object with incidence rate plot.*

Description

Get ggplot object with incidence rate plot.

Usage

```
plotIncidenceRate(
  data,
  cohortIds = NULL,
  databaseIds = NULL,
  stratifyByAgeGroup = TRUE,
  stratifyByGender = TRUE,
  stratifyByCalendarYear = TRUE,
  yscaleFixed = FALSE
)
```

Arguments

data	A tibble data frame object that is the output of getIncidenceRate function.
cohortIds	A vector of one or more integer (bigint) to plot.
databaseIds	A vector of one or more databaseIds to plot.
stratifyByAgeGroup	Do you want to stratify by age?

`stratifyByGender`
: Do you want to stratify by gender?

`stratifyByCalendarYear`
: Do you want to stratify by calendar year?

`yscaleFixed`
: Do you want to rescale y-axis?

Value

A ggplot object.

Examples

```
## Not run:
incidenceRatePlot <- plotIncidenceRate(data = data)

## End(Not run)
```

`plotTimeDistribution` *Get ggplot object with time distribution plot.*

Description

Get ggplot object with time distribution plot.

Usage

```
plotTimeDistribution(
  data,
  cohortIds = NULL,
  databaseIds = NULL,
  xAxis = "database"
)
```

Arguments

`data`
: A tibble data frame object that is the output of `getTimeDistributionResult` function.

`cohortIds`
: A vector of one or more integer (bigint) to plot.

`databaseIds`
: A vector of one or more databaseIds to plot.

`xAxis`
: (optional) By default 'database' will be plotted on x-axis. Alternative is 'cohortId'.

Value

A ggplot object.

Examples

```
## Not run:
timeDistributionPlot <- getTimeDistributionPlot(data = data)

## End(Not run)
```

`preMergeDiagnosticsFiles` *Premerge Shiny diagnostics files*

Description

If there are many diagnostics files, starting the Shiny app may take a very long time. This function already does most of the preprocessing, increasing loading speed.

The merged data will be stored in the same folder, and will automatically be recognized by the Shiny app.

Usage

```
preMergeDiagnosticsFiles(dataFolder, minCovariateProportion = 0)
```

Arguments

`dataFolder`	folder where the exported zip files for the diagnostics are stored. Use the `runCohortDiagnostics` function to generate these zip files. Zip files containing results from multiple databases can be placed in the same folder.
`minCovariateProportion`	minimum value threshold for covariates to be included in premerged file (valid number (maybe decimal) between 0 to 1)

`resolveCohortSqlToConceptIds` *Get a copy of omop vocabulary as csv*

Description

For a given list of conceptId's get a subset of omop vocabulary of these conceptIds. These are written as csv in the export folder

Resolves cohort sql to concept_ids

Usage

```
resolveCohortSqlToConceptIds(
  connection = NULL,
  connectionDetails = NULL,
  cdmDatabaseSchema,
  databaseId,
  oracleTempSchema = NULL,
  cohort
)
```

Arguments

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.
connectionDetails	An object of type connectionDetails as created using the createConnectionDetails function in the DatabaseConnector package. Can be left NULL if connection is provided.
cdmDatabaseSchema	DatabaseSchema where the omop vocabulary files are located.
databaseId	A text string corresponding to the id of the database.
oracleTempSchema	Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.
cohort	A tibble data frame object with atleast two columns. cohortId refering to the integer id that identifies a cohort definition. And sql, which is the cohort definition in OHDSI SQL dialect. It may contain parameters designed to be replaced by SqlRender. The standard form the cohort definition SQL is generated is using circe-be by WebApi and Atlas. The 'cohort' table in Cohort Diagnostics results data model satisfies this requirement.
exportFolder	The folder where the output is exported by Cohort Diagnostics. If this folder does not exist, or does not have the searched file the function will return an error.
conceptIdTable	(optional) A table with one column called concept_id (integer) that contains all the concept_ids to limit the data pull. In the absence of this table, the entire vocabulary is pulled down (slow) vocabulary files. If NULL, all conceptIds are extracted. Please provide the full name of the vocabulary table e.g. databaseSchema.conceptIdTable. If it is a temporary table please use '#conceptIdTable'. Remember, for temporary table the connection object has to be active.
vocabularyTableNames	(optional) A vector of omop vocabulary table names to download.

Value

NULL. The function writes the vocabulary tables into the export folder as csv.

Tibble Data Frame object

---

runCohortDiagnostics *Run cohort diagnostics*

Description

Runs the cohort diagnostics on all (or a subset of) the cohorts instantiated using the `ROhdsiWebApi::insertCohortDefinitionSetInPackage` function. Assumes the cohorts have already been instantiated.

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

```
runCohortDiagnostics(
  packageName = NULL,
  cohortToCreateFile = "settings/CohortsToCreate.csv",
  baseUrl = NULL,
  cohortSetReference = NULL,
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  cohortIds = NULL,
  inclusionStatisticsFolder = NULL,
  exportFolder,
  databaseId,
  databaseName = databaseId,
  databaseDescription = "",
  cdmVersion = 5,
  runInclusionStatistics = TRUE,
  runIncludedSourceConcepts = TRUE,
  runOrphanConcepts = TRUE,
  runTimeDistributions = TRUE,
  runBreakdownIndexEvents = TRUE,
  runIncidenceRate = TRUE,
  runCohortOverlap = TRUE,
  runCohortCharacterization = TRUE,
  covariateSettings = FeatureExtraction::createDefaultCovariateSettings(),
  runTemporalCohortCharacterization = TRUE,

   temporalCovariateSettings = FeatureExtraction::createTemporalCovariateSettings(useConditionOcc
   = TRUE, useDrugEraStart = TRUE, useProcedureOccurrence = TRUE, useMeasurement = TRUE,
   temporalStartDays = c(-365, -30, 0, 1, 31, seq(from = -30, to = -420, by = -30),
   seq(from = 1, to = 390, by = 30)), temporalEndDays = c(-31, -1, 0, 30, 365, seq(from
    = 0, to = -390, by = -30), seq(from = 31, to = 420, by = 30))),
  runResolveCohortSqlToConceptIds = TRUE,
  runCombineConceptSetsFromCohorts = TRUE,
  minCellCount = 5,
  incremental = FALSE,
  incrementalFolder = exportFolder
)
```

Arguments

packageName	The name of the package containing the cohort definitions. Can be left NULL if baseUrl and cohortSetReference have been specified.
cohortToCreateFile	The location of the cohortToCreate file within the package. Is ignored if baseUrl and cohortSetReference have been specified. The cohortToCreateFile must be .csv file that is expected to be read into a dataframe object identical to requirements for cohortSetReference argument.
baseUrl	The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Can be left NULL if packageName and cohortToCreateFile have been specified.
cohortSetReference	A data frame with four columns, as described in the details. Can be left NULL if packageName and cohortToCreateFile have been specified.
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'.
oracleTempSchema	Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.
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	Optionally, provide a subset of cohort IDs to restrict the diagnostics to.
inclusionStatisticsFolder	The folder where the inclusion rule statistics are stored. Can be left NULL if runInclusionStatistics = FALSE.
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').
databaseName	The full name of the database.
databaseDescription	A short description (several sentences) of the database.
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?

`runTimeDistributions`
: Generate and export cohort time distributions?

`runBreakdownIndexEvents`
: Generate and export the breakdown of index events?

`runIncidenceRate`
: Generate and export the cohort incidence rates?

`runCohortOverlap`
: Generate and export the cohort overlap? Overlaps are checked within cohortIds that have the same referrent conceptId sourced from the CohortSetReference or cohortToCreateFile.

`runCohortCharacterization`
: Generate and export the cohort characterization? Only records with values greater than 0.0001 are returned.

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

`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.

`runResolveCohortSqlToConceptIds`
: Resolve and export all the concept ids in all the concept set expressions in cohorts?

`runCombineConceptSetsFromCohorts`
: Generate and export all the concept set expressions from cohorts?

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

`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.

`vocabularyDatabaseSchema`
: (optional) Schema name where your vocabulary resides. Most commonly it is the same as CDM databaseSchema. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.

Details

Currently two ways of executing this function are supported, either (1) [Package Mode] embedded in a study package, assuming the cohort definitions are stored in that package using the `ROhdsiWebApi::insertCohortDefinitionSetInPackage`, or (2) [WebApi Mode] By using a WebApi interface to retrieve the cohort definitions.

When using this function in Package Mode: Use the `packageName` and `cohortToCreateFile` to specify the name of the study package, and the name of the cohortToCreate file within that package, respectively

When using this function in WebApi Mode: use the `baseUrl` and `cohortSetReference` to specify how to connect to the WebApi, and which cohorts to fetch, respectively.

Note: if the parameters for both Package Mode and WebApi Mode are provided, then Package mode is preferred.

The `cohortSetReference` argument must be a data frame with the following columns:

referentConceptId A standard omop concept id that serves as the referant phenotype definition for the cohort Id.

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. It is recommended to be (referrentConceptId * 1000) + a number between 3 to 999

webApiCohortId Cohort Id in the webApi/atlas instance. It is a required field to run Cohort Diagnostics in WebApi mode. It is discarded in package mode.

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

logicDescription A human understandable brief description of the cohort definition. This logic does not have to a fully specified description of the cohort definition, but should provide enough context to help user understand the meaning of the cohort definition

---

`runCohortDiagnosticsUsingExternalCounts`
Run cohort diagnostics using external concept counts

Description

Runs cohort diagnostics on all (or a subset of) the cohorts, but using external concept counts. The external counts must have the following columns:

concept_id The source or target concept ID.

concept_count The number of records having the concept.

concept_subjects The number of unique persons having the concept.

Usage

```
runCohortDiagnosticsUsingExternalCounts(
  packageName = NULL,
  cohortToCreateFile = "settings/CohortsToCreate.csv",
  baseUrl = NULL,
  cohortSetReference = NULL,
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  cohortIds = NULL,
  conceptCountsDatabaseSchema = cdmDatabaseSchema,
  conceptCountsTable = "concept_counts",
  conceptCountsTableIsTemp = FALSE,
```

```
  exportFolder,
  databaseId,
  databaseName = databaseId,
  databaseDescription = "",
  runIncludedSourceConcepts = TRUE,
  runOrphanConcepts = TRUE,
  minCellCount = 5
)
```

Arguments

packageName	The name of the package containing the cohort definitions. Can be left NULL if baseUrl and cohortSetReference have been specified.
cohortToCreateFile	The location of the cohortToCreate file within the package. Is ignored if baseUrl and cohortSetReference have been specified. The cohortToCreateFile must be .csv file that is expected to be read into a dataframe object identical to requirements for cohortSetReference argument.
baseUrl	The base URL for the WebApi instance, for example: "http://server.org:80/WebAPI". Can be left NULL if packageName and cohortToCreateFile have been specified.
cohortSetReference	A data frame with four columns, as described in the details. Can be left NULL if packageName and cohortToCreateFile have been specified.
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'.
oracleTempSchema	Should be used in Oracle to specify a schema where the user has write privileges for storing temporary tables.
cohortIds	Optionally, provide a subset of cohort IDs to restrict the diagnostics to.
conceptCountsDatabaseSchema	Schema name where your concept counts table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'. Ignored if conceptCountsTableIsTemp = TRUE.
conceptCountsTable	Name of the concept counts table. This table can be created using the createConceptCountsTable.
conceptCountsTableIsTemp	Is the concept counts table a temp table?
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').

`databaseName` The full name of the database.

`databaseDescription`
A short description (several sentences) of the database.

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

`runOrphanConcepts`
Generate and export potential orphan concepts?

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

Details

Currently two ways of executing this function are supported, either (1) [Package Mode] embedded in a study package, assuming the cohort definitions are stored in that package using the `ROhdsiWebApi::insertCohortDefinitionSetInPackage`, or (2) [WebApi Mode] By using a WebApi interface to retrieve the cohort definitions.

When using this function in Package Mode: Use the `packageName` and `cohortToCreateFile` to specify the name of the study package, and the name of the cohortToCreate file within that package, respectively

When using this function in WebApi Mode: use the `baseUrl` and `cohortSetReference` to specify how to connect to the WebApi, and which cohorts to fetch, respectively.

Note: if the parameters for both Package Mode and WebApi Mode are provided, then Package mode is preferred.

The `cohortSetReference` argument must be a data frame with the following columns:

referentConceptId A standard omop concept id that serves as the referant phenotype definition for the cohort Id.

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. It is recommended to be (referrentConceptId * 1000) + a number between 3 to 999

webApiCohortId Cohort Id in the webApi/atlas instance. It is a required field to run Cohort Diagnostics in WebApi mode. It is discarded in package mode.

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

logicDescription A human understandable brief description of the cohort definition. This logic does not have to a fully specified description of the cohort definition, but should provide enough context to help user understand the meaning of the cohort definition

Index