

# Package ‘CohortDiagnostics’

May 3, 2022

**Type** Package

**Title** Diagnostics for OHDSI Cohorts

**Version** 3.0.0

**Date** 2022-04-27

**Maintainer** Jamie Gilbert <gilbert@ohdsi.org>

**Description**

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

**Depends** DatabaseConnector (*i*= 5.0.0),

FeatureExtraction (*i*= 3.2.0),

R (*i*= 4.1.0)

**Imports** Andromeda,

checkmate,

clock,

digest,

dplyr (*i*= 1.0.0),

methods,

ParallelLogger (*i*= 3.0.0),

readr (*i*= 2.1.0),

RJSONIO,

rlang,

SqlRender (*i*= 1.9.0),

stringr,

tidyr (*i*= 1.2.0),

CohortGenerator (*i*= 0.4.0)

**Suggests** CirceR,

DT,

Eunomia,

ggiraph,

ggplot2,

htmltools,

knitr,

lubridate,

pool,

plotly,

purrr,

RColorBrewer,

remotes,

rmarkdown,

ROhdsiWebApi (*i*= 1.2.0),  
 RSQLite (*i*= 2.2.1),  
 scales,  
 shiny,  
 shinydashboard,  
 shinyWidgets,  
 testthat,  
 withr,  
 zip

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

**License** Apache License

**VignetteBuilder** knitr

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

**BugReports** <https://github.com/OHDSI/CohortDiagnostics/issues>

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**Language** en-US

## R topics documented:

checkInputFileEncoding . . . . .	3
createDiagnosticsExplorerZip . . . . .	3
createMergedResultsFile . . . . .	4
createResultsDataModel . . . . .	4
executeDiagnostics . . . . .	5
getCdmDataSourceInformation . . . . .	9
getCohortCounts . . . . .	10
getDefaultVocabularyTableNames . . . . .	11
getResultsDataModelSpecifications . . . . .	11
launchCohortExplorer . . . . .	12
launchDiagnosticsExplorer . . . . .	13
loadCohortsFromPackage . . . . .	14
runCohortRelationshipDiagnostics . . . . .	14
runCohortTimeSeriesDiagnostics . . . . .	16
takepackageDependencySnapshot . . . . .	17
uploadResults . . . . .	18

---

checkInputFileEncoding

*Check character encoding of input file*


---

### Description

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

### Usage

```
checkInputFileEncoding(fileName)
```

### Arguments

fileName            The path to the file to check

### Value

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

---

createDiagnosticsExplorerZip

*Create publishable shiny zip*


---

### Description

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

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

### Usage

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

### Arguments

outputZipfile    The output path for the zip file

sqliteDbPath    Merged Cohort Diagnostics sqlitedb created with [createMergedResultsFile](#)

shinyDirectory   (optional) Path to the location where the shiny code is stored. By default, this is the package root

overwrite        If the zip file already exists, overwrite it?

---

**createMergedResultsFile**

*Merge Shiny diagnostics files into sqlite database*

---

**Description**

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

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

**Usage**

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

**Arguments**

<b>dataFolder</b>	folder where the exported zip files for the diagnostics are stored. Use the <a href="#">executeDiagnostics</a> function to generate these zip files. Zip files containing results from multiple databases may be placed in the same folder.
<b>sqliteDbPath</b>	Output path where sqlite database is placed
<b>overwrite</b>	(Optional) overwrite existing sqlite lite db if it exists.

---

**createResultsDataModel**

*Create the results data model tables on a database server.*

---

**Description**

Create the results data model tables on a database server.

**Usage**

```
createResultsDataModel(connection = NULL, connectionDetails = NULL, schema)
```

**Arguments**

<b>connection</b>	An object of type <code>connection</code> as created using the <a href="#">connect</a> function in the DatabaseConnector package. Can be left NULL if <code>connectionDetails</code> is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
<b>connectionDetails</b>	An object of type <code>connectionDetails</code> as created using the <a href="#">createConnectionDetails</a> function in the DatabaseConnector package. Can be left NULL if <code>connection</code> is provided.
<b>schema</b>	The schema on the postgres server where the tables will be created.

## Details

Only PostgreSQL servers are supported.

---

executeDiagnostics	<i>Execute cohort diagnostics</i>
--------------------	-----------------------------------

---

## Description

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

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

## Usage

```
<<<<<<< HEAD:man/executeDiagnostics.Rd
executeDiagnostics(
  cohortDefinitionSet,
  exportFolder,
  databaseId,
  cohortDatabaseSchema,
  databaseName = databaseId,
  databaseDescription = databaseId,
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cohortTable = "cohort",
  cohortTableNames = CohortGenerator::getCohortTableNames(cohortTable = cohortTable),
  vocabularyDatabaseSchema = cdmDatabaseSchema,
  cohortIds = NULL,
  =====
runCohortDiagnostics(
  packageName = NULL,
  cohortToCreateFile = "settings/CohortsToCreate.csv",
  cohortDefinitionSet = NULL,
  baseUrl = NULL,
  cohortSetReference = NULL,
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  oracleTempSchema = NULL,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cohortDatabaseSchema,
  vocabularyDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  cohortTableNames = CohortGenerator::getCohortTableNames(cohortTable = cohortTable),
  cohortIds = NULL,
  inclusionStatisticsFolder = NULL,
```

```

exportFolder,
databaseId,
databaseName = databaseId,
databaseDescription = databaseId,
>>>>>>> main:man/runCohortDiagnostics.Rd
  cdmVersion = 5,
  runInclusionStatistics = TRUE,
  runIncludedSourceConcepts = TRUE,
  runOrphanConcepts = TRUE,
  runTimeSeries = FALSE,
  runVisitContext = TRUE,
  runBreakdownIndexEvents = TRUE,
  runIncidenceRate = TRUE,
  runCohortRelationship = TRUE,
  runTemporalCohortCharacterization = TRUE,

  temporalCovariateSettings = FeatureExtraction::createTemporalCovariateSettings(useDemographics
    = TRUE, useDemographicsAge = TRUE, useDemographicsAgeGroup = TRUE,
    useDemographicsRace = TRUE, useDemographicsEthnicity = TRUE, useDemographicsIndexYear
    = TRUE, useDemographicsIndexMonth = TRUE, useDemographicsIndexYearMonth = TRUE,
    useDemographicsPriorObservationTime = TRUE, useDemographicsPostObservationTime =
    TRUE, useDemographicsTimeInCohort = TRUE, useConditionOccurrence = TRUE,
    useProcedureOccurrence = TRUE, useDrugEraStart = TRUE, useMeasurement = TRUE,
    useConditionEraStart = TRUE, useConditionEraOverlap = TRUE, useConditionEraGroupStart
    = FALSE, useConditionEraGroupOverlap = TRUE, useDrugExposure = FALSE,
    useDrugEraOverlap = FALSE, useDrugEraGroupStart = FALSE, useDrugEraGroupOverlap =
    TRUE, useObservation = TRUE, useDeviceExposure = TRUE, useCharlsonIndex = TRUE,
    useDcsi = TRUE, useChads2 = TRUE, useChads2Vasc = TRUE, useHfrs = FALSE,
    temporalStartDays = c(-9999, -365, -180, -30, -365, -30, 0, 1, 31, -9999),
    temporalEndDays = c(0, 0, 0, 0, -31, -1, 0, 30, 365, 9999)),
  minCellCount = 5,
  incremental = FALSE,
  incrementalFolder = file.path(exportFolder, "incremental")
)

```

## Arguments

cohortDefinitionSet	Data.frame of cohorts must include columns cohortId, cohortName, json, sql
exportFolder	The folder where the output will be exported to. If this folder does not exist it will be created.
databaseId	A short string for identifying the database (e.g. 'Synpuf'). iiiiiii HEAD:man/executeDiagnostics.Rd
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'. =====
cohortDefinitionSet	Data.frame of cohorts must include columns cohortId, cohortName, json, sql

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. <i>llllllll</i> main:man/runCohortDiagnostics.Rd
databaseName	The full name of the database. If NULL, defaults to databaseId.
databaseDescription	A short description (several sentences) of the database. If NULL, defaults to databaseId.
connectionDetails	An object of type connectionDetails as created using the <a href="#">createConnectionDetails</a> function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type connection as created using the <a href="#">connect</a> function in the DatabaseConnector package. Can be left NULL if connectionDetails is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.
cdmDatabaseSchema	Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.
tempEmulationSchema	Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
cohortTable	Name of the cohort table.
cohortTableNames	Cohort Table names used by CohortGenerator package
vocabularyDatabaseSchema	Schema name where your OMOP vocabulary data resides. This is commonly the same as cdmDatabaseSchema. Note that for SQL Server, this should include both the database and schema name, for example 'vocabulary.dbo'.
	<i>iiiiiii</i> HEAD:man/executeDiagnostics.Rd =====
cohortTable	Name of the cohort table.
cohortTableNames	Cohort Table names used by CohortGenerator package
	<i>llllllll</i> main:man/runCohortDiagnostics.Rd
cohortIds	Optionally, provide a subset of cohort IDs to restrict the diagnostics to.
cdmVersion	The version of the OMOP CDM. Default 5. (Note: only 5 is supported.)
runInclusionStatistics	Generate and export statistic on the cohort inclusion rules?
runIncludedSourceConcepts	Generate and export the source concepts included in the cohorts?
runOrphanConcepts	Generate and export potential orphan concepts?
runTimeSeries	Generate and export the time series diagnostics?
runVisitContext	Generate and export index-date visit context?
runBreakdownIndexEvents	Generate and export the breakdown of index events?

```

runIncidenceRate
    Generate and export the cohort incidence rates?
runCohortRelationship
    Generate and export the cohort relationship? Cohort relationship checks
    the temporal relationship between two or more cohorts.
runTemporalCohortCharacterization
    Generate and export the temporal cohort characterization? Only records
    with values greater than 0.001 are returned.
temporalCovariateSettings
    Either an object of type covariateSettings as created using one of the
    createTemporalCovariateSettings function in the FeatureExtraction pack-
    age, or a list of such objects.
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.

```

## Details

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

```
##### HEAD:man/executeDiagnostics.Rd
```

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

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

**json** The JSON cohort definition for the cohort.

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

```

===== \itemcohortIdThe 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 (referentConceptId * 1000) + a number between 3 to 999 \itematlasIdCohort Id
in the webApi/atlas instance. It is a required field to run Cohort Diagnostics in We-
bApi mode. It is discarded in package mode. \itemcohortNameThe full name of the co-
hort. This will be shown in the Shiny app. \itemlogicDescriptionA 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 un-
derstand the meaning of the cohort definition \itemreferentConceptIdA standard omop
concept id that serves as the referent phenotype definition for the cohort Id (optional)
##### main:man/runCohortDiagnostics.Rd

```

## Examples

```

## Not run:
# Load cohorts (assumes that they have already been instantiated)
cohortTableNames <- CohortGenerator::getCohortTableNames(cohortTable = "cohort")
cohorts <- CohortGenerator::getCohortDefinitionSet(packageName = "MyGreatPackage")
connectionDetails <- createConnectionDetails(
  dbms = "postgresql",

```



```
server = "ohdsi.com",
port = 5432,
user = "me",
password = "secure"
)

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

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

## End(Not run)
```

---

**getCdmDataSourceInformation**

*Returns information from CDM source table.*

---

## Description

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

## Usage

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

**Arguments**

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

**Value**

Returns a data frame from CDM Data source.

---

getCohortCounts	<i>Count the cohort(s)</i>
-----------------	----------------------------

---

**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 <code>connectionDetails</code> as created using the <a href="#">createConnectionDetails</a> function in the DatabaseConnector package. Can be left NULL if connection is provided.
connection	An object of type <code>connection</code> as created using the <a href="#">connect</a> function in the DatabaseConnector package. Can be left NULL if <code>connectionDetails</code> 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

---

`getDefaultVocabularyTableNames`

*Get a list of vocabulary table names*

---

**Description**

Get a list of vocabulary table names

**Usage**

`getDefaultVocabularyTableNames()`

**Value**

Get a list of vocabulary table names in results data model

---

`getResultsDataModelSpecifications`

*Get specifications for Cohort Diagnostics results data model*

---

**Description**

Get specifications for Cohort Diagnostics results data model

**Usage**

`getResultsDataModelSpecifications()`

**Value**

A tibble data frame object with specifications

---

launchCohortExplorer	<i>Launch the CohortExplorer Shiny app</i>
----------------------	--

---

## Description

Launch the CohortExplorer Shiny app

## Usage

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

## Arguments

connectionDetails	An object of type <code>connectionDetails</code> as created using the <a href="#">createConnectionDetails</a> 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 <code>subjectIds</code> 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(
  sqliteDbPath = "MergedCohortDiagnosticsData.sqlite",
  connectionDetails = NULL,
  resultsDatabaseSchema = NULL,
  vocabularyDatabaseSchema = NULL,
  vocabularyDatabaseSchemas = resultsDatabaseSchema,
  aboutText = NULL,
  runOverNetwork = FALSE,
  port = 80,
  launch.browser = FALSE
)
```

## Arguments

- sqliteDbPath** Path to merged sqlite file. See [createMergedResultsFile](#) to create file.
- connectionDetails** An object of type `connectionDetails` as created using the [createConnectionDetails](#) function in the `DatabaseConnector` package, specifying how to connect to the server where the CohortDiagnostics results have been uploaded using the [uploadResults](#) function.
- resultsDatabaseSchema** The schema on the database server where the CohortDiagnostics results have been uploaded.
- vocabularyDatabaseSchema** (Deprecated) Please use `vocabularyDatabaseSchemas`.
- vocabularyDatabaseSchemas** (optional) A list of one or more schemas on the database server where the vocabulary tables are located. The default value is the value of the `resultsDatabaseSchema`. We can provide a list of vocabulary schema that might represent different versions of the OMOP vocabulary tables. It allows us to compare the impact of vocabulary changes on Diagnostics. Not supported with an sqlite database.
- aboutText** Text (using HTML markup) that will be displayed in an About tab in the Shiny app. If not provided, no About tab will be shown.
- runOverNetwork** (optional) Do you want the app to run over your network?
- port** (optional) Only used if `runOverNetwork = TRUE`.
- launch.browser** Should the app be launched in your default browser, or in a Shiny window. Note: copying to clipboard will not work in a Shiny window.

## Details

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

---

loadCohortsFromPackage

*Load Cohort Definitions From A Study Package*

---

## Description

Load cohort references for usage in executeDiagnostics.

## Usage

```
loadCohortsFromPackage(
  packageName,
  cohortToCreateFile = "settings/cohortsToCreate.csv",
  cohortIds = NULL,
  errorMessage = 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. This csv file is expected to be encoded in either ASCII or UTF-8, if not, an error message will be displayed and process stopped.
cohortIds	Optionally, provide a subset of cohort IDs to restrict the diagnostics to.
errorMessage	checkmate assert collection, used internally for error checks

---

runCohortRelationshipDiagnostics

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

---

## Description

Given a set of cohorts, get temporal relationships between the cohort\_start\_date of the cohorts.

## Usage

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

## 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'.
- cdmDatabaseSchema**  
Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm\_data.dbo'.
- tempEmulationSchema**  
Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.
- cohortTable**  
Name of the cohort table.
- targetCohortIds**  
A vector of one or more Cohort Ids for use as target cohorts.
- comparatorCohortIds**  
A vector of one or more Cohort Ids for use as feature/comparator cohorts.
- relationshipDays**  
A dataframe with two columns `startDay` and `endDay` representing periods of time to compute relationship
- observationPeriodRelationship**  
Do you want to compute temporal relationship between target cohort and observation period table?

---

**runCohortTimeSeriesDiagnostics**

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

---

**Description**

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

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

**Usage**

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

**Arguments**

- |                                  |  |
|----------------------------------|--|
| <code>connectionDetails</code>   | An object of type <code>connectionDetails</code> as created using the <a href="#">createConnectionDetails</a> function in the DatabaseConnector package. Can be left NULL if <code>connection</code> is provided.  |
| <code>connection</code>          | An object of type <code>connection</code> as created using the <a href="#">connect</a> function in the DatabaseConnector package. Can be left NULL if <code>connectionDetails</code> is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes. |
| <code>tempEmulationSchema</code> | Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.  |
| <code>cdmDatabaseSchema</code>   | Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.   |



cohortDatabaseSchema	Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
cohortTable	Name of the cohort table.
runCohortTimeSeries	Generate and export the cohort level time series?
runDataSourceTimeSeries	Generate and export the Data source level time series? i.e. using all persons found in observation period table.
timeSeriesMinDate	(optional) Minimum date for time series. Default value January 1st 1980.
timeSeriesMaxDate	(optional) Maximum date for time series. Default value System date.
stratifyByGender	Do you want to stratify by Gender
stratifyByAgeGroup	Do you want to stratify by Age group
cohortIds	A vector of one or more Cohort Ids to compute time distribution for.

---

takepackageDependencySnapshot

*Take a snapshot of the R environment*


---

## Description

Take a snapshot of the R environment

## Usage

```
takepackageDependencySnapshot()
```

## Details

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

## Value

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

---

uploadResults	<i>Upload results to the database server.</i>
---------------	---

---

## Description

Requires the results data model tables have been created using the [createResultsDataModel](#) function.

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

## Usage

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

## Arguments

connectionDetails	An object of type <code>connectionDetails</code> as created using the <a href="#">createConnectionDetails</a> function in the DatabaseConnector package.
schema	The schema on the postgres server where the tables have been created.
zipFileName	The name of the zip file.
forceOverWriteOfSpecifications	If TRUE, specifications of the phenotypes, cohort definitions, and analysis will be overwritten if they already exist on the database. Only use this if these specifications have changed since the last upload.
purgeSiteDataBeforeUploading	If TRUE, before inserting data for a specific databaseId all the data for that site will be dropped. This assumes the input zip file contains the full data for that data site.
tempFolder	A folder on the local file system where the zip files are extracted to. Will be cleaned up when the function is finished. Can be used to specify a temp folder on a drive that has sufficient space if the default system temp space is too limited.