Package ‘CohortGenerator’

May 25, 2022

Type Package

Title An R Package for Cohort Generation Against the OMOP CDM

Version 0.5.0

Date 2022-05-25

Maintainer Anthony Sena <sena@ohdsi.org>

Description
An R package for that encapsulates the functions for generating cohorts against the OMOP CDM.

Depends DatabaseConnector (>= 5.0.0),
R (>= 3.6.0)

Imports checkmate,
digest,
dplyr,
lubridate,
ParallelLogger (>= 3.0.0),
readr (>= 2.1.0),
rlang,
RJSONIO,
SqlRender (>= 1.7.0),
stringi (>= 1.7.6)

Suggests CirceR (>= 1.1.1),
Eunomia,
knitr,
rmarkdown,
ROhdsiWebApi,
testthat

Remotes ohdsi/CirceR,
ohdsi/Eunomia,
ohdsi/ROhdsiWebApi

License Apache License

VignetteBuilder knitr

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

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

RoxygenNote 7.1.2

Encoding UTF-8

Language en-US

R topics documented:

`.readCsv` *Internal read CSV file when control over column formatting is required.*

Description

This function is internal to the package

Usage

```
.readCsv(file)
```

Arguments

`file`	The .csv file to read.

Value

Returns the file contents invisibly.

`.writeCsv` *Internal write.csv file when control over column formatting is required.*

Description

This function is internal to the package since the `exportCohortStatsTables` requires additional control over the column formatting which requires that we bypass the `writeCsv` function since it will automatically convert from camelCase to snake_case.

Usage

```
.writeCsv(x, file)
```

Arguments

`x`	A data frame or tibble to write to disk.
`file`	The .csv file to write.

Value

Returns the input x invisibly.

`checkSettingsColumns` *Custom checkmate assertion for ensuring the settings columns are properly specified*

Description

This function is used to provide a more informative message when ensuring that the columns in the cohort definition set or the CSV file that defines the cohort definition set is properly specified. This function is then bootstrapped upon package initialization (code in CohortGenerator.R) to allow for it to work with the other checkmate assertions as described in: https://mllg.github.io/checkmate/articles/checkmate.html. The assertion function is called assert_settings_columns.

Usage

```
checkSettingsColumns(columnNames, settingsFileName = NULL)
```

Arguments

`columnNames`	The name of the columns found in either the cohortDefinitionSet data frame or from reading the contents of the settingsFile
`settingsFileName`	The file name of the CSV that defines the cohortDefinitionSet. When NULL, this function assumes the column names are defined in a data.frame representation of the cohortDefinitionSet

Value

Returns TRUE if all required columns are found otherwise it returns an error

computeChecksum *Computes the checksum for a value*

Description

This is used as part of the incremental operations to hash a value to store in a record keeping file. This function leverages the md5 hash from the digest package

Usage

```
computeChecksum(val)
```

Arguments

`val`	The value to hash. It is converted to a character to perform the hash.

Value

Returns a string containing the checksum

createCohortTables *Create cohort tables*

Description

This function creates an empty cohort table and empty tables for cohort statistics.

Usage

```
createCohortTables(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTableNames = getCohortTableNames(),
  incremental = 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.
`cohortDatabaseSchema`	Schema name where your cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortTableNames`	The names of the cohort tables. See `getCohortTableNames` for more details.
`incremental`	When set to TRUE, this function will check to see if the cohortTableNames exists in the cohortDatabaseSchema and if they exist, it will skip creating the tables.

`createEmptyCohortDefinitionSet` *Create an empty cohort definition set*

Description

This function creates an empty cohort set data.frame for use with `generateCohortSet`.

Usage

```
createEmptyCohortDefinitionSet()
```

Value

Returns an empty cohort set data.frame

`dropCohortStatsTables` *Drop cohort statistics tables*

Description

This function drops the cohort statistics tables.

Usage

```
dropCohortStatsTables(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTableNames = getCohortTableNames()
)
```

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

`cohortTableNames`
: The names of the cohort tables. See `getCohortTableNames` for more details.

`exportCohortStatsTables` *Export the cohort statistics tables to the file system*

Description

This function retrieves the data from the cohort statistics tables and writes them to the inclusion statistics folder specified in the function call.

Usage

```
exportCohortStatsTables(
  connectionDetails,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTableNames = getCohortTableNames(),
  cohortStatisticsFolder,
  snakeCaseToCamelCase = TRUE,
  fileNamesInSnakeCase = FALSE,
  incremental = FALSE,
  databaseId = 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.

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

`cohortTableNames`
: The names of the cohort tables. See `getCohortTableNames` for more details.

`cohortStatisticsFolder`
: The path to the folder where the cohort statistics folder where the results will be written

`snakeCaseToCamelCase`
: Should column names in the exported files convert from snake_case to camelCase? Default is FALSE

`fileNamesInSnakeCase`	Should the exported files use snake_case? Default is FALSE
`incremental`	If `incremental = TRUE`, results are written to update values instead of overwriting an existing results
`databaseId`	Optional - when specified, the databaseId will be added to the exported results

---

`generateCohort` *Generates a cohort*

Description

This function is used by `generateCohortSet` to generate a cohort against the CDM.

Usage

```
generateCohort(
  cohortId = NULL,
  cohortDefinitionSet,
  connection = NULL,
  connectionDetails = NULL,
  cdmDatabaseSchema,
  tempEmulationSchema,
  cohortDatabaseSchema,
  cohortTableNames,
  stopIfError = TRUE,
  incremental,
  recordKeepingFile
)
```

Arguments

`cohortId` The cohortId in the list of `cohortDefinitionSet`

`cohortDefinitionSet`

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

cohortId The unique integer identifier of the cohort

cohortName The cohort's name

sql The OHDSI-SQL used to generate the cohort

Optionally, this data frame may contain:

json The Circe JSON representation of the cohort

`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`
: 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.

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

`cohortTableNames`
: The names of the cohort tables. See `getCohortTableNames` for more details.

`stopIfError`
: When set to true, an error in processing will call the stop() command to notify the parent calling function that an error occurred.

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

`recordKeepingFile`
: If `incremental` = `TRUE`, this file will contain information on cohorts already generated

---

`generateCohortSet` *Generate a set of cohorts*

Description

This function generates a set of cohorts in the cohort table.

Usage

```
generateCohortSet(
  connectionDetails = NULL,
  connection = NULL,
  cdmDatabaseSchema,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTableNames = getCohortTableNames(),
  cohortDefinitionSet = NULL,
  stopOnError = TRUE,
  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'.
`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.
`cohortDatabaseSchema`	Schema name where your cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`cohortTableNames`	The names of the cohort tables. See `getCohortTableNames` for more details.
`cohortDefinitionSet`	The `cohortDefinitionSet` argument must be a data frame with the following columns: **cohortId** The unique integer identifier of the cohort **cohortName** The cohort's name **sql** The OHDSI-SQL used to generate the cohort Optionally, this data frame may contain: **json** The Circe JSON representation of the cohort
`stopOnError`	If an error happens while generating one of the cohorts in the cohortDefinitionSet, should we stop processing the other cohorts? The default is TRUE; when set to FALSE, failures will be identified in the return value from this function.
`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.

Value

A data.frame consisting of the following columns:

cohortId The unique integer identifier of the cohort

cohortName The cohort's name

generationStatus The status of the generation task which may be one of the following:

- **COMPLETE** The generation completed successfully
- **FAILED** The generation failed (see logs for details)
- **SKIPPED** If using incremental == 'TRUE', this status indicates that the cohort's generation was skipped since it was previously completed.

startTime The start time of the cohort generation. If the generationStatus == 'SKIPPED', the startTime will be NA.

endTime The end time of the cohort generation. If the generationStatus == 'FAILED', the endTime will be the time of the failure. If the generationStatus == 'SKIPPED', endTime will be NA.

getCohortCounts *Count the cohort(s)*

Description

Computes the subject and entry count per cohort. Note the cohortDefinitionSet parameter is optional - if you specify the cohortDefinitionSet, the cohort counts will be joined to the cohortDefinitionSet to include attributes like the cohortName.

Usage

```
getCohortCounts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  cohortIds = c(),
  cohortDefinitionSet = NULL,
  databaseId = 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.
`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`	The 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.
`cohortDefinitionSet`	The `cohortDefinitionSet` argument must be a data frame with the following columns: **cohortId** The unique integer identifier of the cohort; **cohortName** The cohort's name; **sql** The OHDSI-SQL used to generate the cohort. Optionally, this data frame may contain: **json** The Circe JSON representation of the cohort
`databaseId`	Optional - when specified, the databaseId will be added to the exported results

Value

A data frame with cohort counts

getCohortDefinitionSet

Get a cohort definition set

Description

This function supports the legacy way of retrieving a cohort definition set from the file system or in a package. This function supports the legacy way of storing a cohort definition set in a package with a CSV file, JSON files, and SQL files in the 'inst' folder.

Usage

```
getCohortDefinitionSet(
  settingsFileName = "Cohorts.csv",
  jsonFolder = "cohorts",
  sqlFolder = "sql/sql_server",
  cohortFileNameFormat = "%s",
  cohortFileNameValue = c("cohortId"),
  packageName = NULL,
  warnOnMissingJson = TRUE,
  verbose = FALSE
)
```

Arguments

`settingsFileName`	The name of the CSV file that will hold the cohort information including the cohortId and cohortName
`jsonFolder`	The name of the folder that will hold the JSON representation of the cohort if it is available in the cohortDefinitionSet
`sqlFolder`	The name of the folder that will hold the SQL representation of the cohort.
`cohortFileNameFormat`	Defines the format string for naming the cohort JSON and SQL files. The format string follows the standard defined in the base sprintf function.
`cohortFileNameValue`	Defines the columns in the cohortDefinitionSet to use in conjunction with the cohortFileNameFormat parameter.
`packageName`	The name of the package containing the cohort definitions.
`warnOnMissingJson`	Provide a warning if a .JSON file is not found for a cohort in the settings file
`verbose`	When TRUE, extra logging messages are emitted

Value

Returns a cohort set data.frame

`getCohortStats` *Get Cohort Inclusion Stats Table Data*

Description

This function returns a data frame of the data in the Cohort Inclusion Tables. Results are organized in to a list with 5 different data frames:

- cohortInclusionTable
- cohortInclusionResultTable
- cohortInclusionStatsTable
- cohortSummaryStatsTable
- cohortCensorStatsTable

These can be optionally specified with the `outputTables`. See `exportCohortStatsTables` function for saving data to csv.

Usage

```
getCohortStats(
  connectionDetails,
  connection = NULL,
  cohortDatabaseSchema,
  databaseId = NULL,
  snakeCaseToCamelCase = TRUE,
  outputTables = c("cohortInclusionTable", "cohortInclusionResultTable",
   "cohortInclusionStatsTable", "cohortInclusionStatsTable", "cohortSummaryStatsTable",
    "cohortCensorStatsTable"),
  cohortTableNames = getCohortTableNames()
)
```

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 tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
`databaseId`	Optional - when specified, the databaseId will be added to the exported results
`snakeCaseToCamelCase`	Convert column names from snake case to camel case.

`outputTables`	Character vector. One or more of "cohortInclusionTable", "cohortInclusionResultTable", "cohortInclusionStatsTable", "cohortInclusionStatsTable", "cohortSummaryStatsTable" or "cohortCensorStatsTable". Output is limited to these tables. Cannot export, for, example, the cohort table. Defaults to all stats tables.
`cohortTableNames`	The names of the cohort tables. See `getCohortTableNames` for more details.

`getCohortTableNames` *Used to get a list of cohort table names to use when creating the cohort tables*

Description

This function creates a list of table names used by `createCohortTables` to specify the table names to create. Use this function to specify the names of the main cohort table and cohort statistics tables.

Usage

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

Arguments

`cohortTable`	Name of the cohort table.
`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.
`cohortCensorStatsTable`	Name of the censor stats table, one of the tables for storing inclusion rule statistics.

Value

A list of the table names as specified in the parameters to this function.

getRequiredTasks — *Get a list of tasks required when running in incremental mode*

Description

This function will attempt to check the `recordKeepingFile` to determine if a list of operations have completed by comparing the keys passed into the function with the checksum supplied

Usage

```
getRequiredTasks(..., checksum, recordKeepingFile)
```

Arguments

`...`	Parameter values used to identify the key in the incremental record keeping file
`checksum`	The checksum representing the operation to check
`recordKeepingFile`	A file path to a CSV file containing the record keeping information.

Value

Returns a list of outstanding tasks based on inspecting the full contents of the record keeping file

insertInclusionRuleNames — *Used to insert the inclusion rule names from a cohort definition set when generating cohorts that include cohort statistics*

Description

This function will take a cohortDefinitionSet that inclusions the Circe JSON representation of each cohort, parse the InclusionRule property to obtain the inclusion rule name and sequence number and insert the values into the cohortInclusionTable. This function is only required when generating cohorts that include cohort statistics.

Usage

```
insertInclusionRuleNames(
  connectionDetails = NULL,
  connection = NULL,
  cohortDefinitionSet,
  cohortDatabaseSchema,
  cohortInclusionTable = getCohortTableNames()$cohortInclusionTable
)
```

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.
`cohortDefinitionSet`	The `cohortDefinitionSet` argument must be a data frame with the following columns: **cohortId** The unique integer identifier of the cohort; **cohortName** The cohort's name; **sql** The OHDSI-SQL used to generate the cohort. Optionally, this data frame may contain: **json** The Circe JSON representation of the cohort
`cohortDatabaseSchema`	Schema name where your cohort 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.

Value

A data frame containing the inclusion rules by cohort and sequence ID

`isCamelCase` *Used to check if a string is in lower camel case*

Description

This function is used check if a string conforms to the lower camel case format.

Usage

```
isCamelCase(x)
```

Arguments

`x`	The string to evaluate

Value

TRUE if the string is in lower camel case

`isFormattedForDatabaseUpload` *Is the data.frame formatted for uploading to a database?*

Description

This function is used to check a data.frame to ensure all column names are in snake case format.

Usage

```
isFormattedForDatabaseUpload(x, warn = TRUE)
```

Arguments

`x`	A data frame
`warn`	When TRUE, display a warning of any columns are not in snake case format

Value

Returns TRUE if all columns are snake case format. If warn == TRUE, the function will emit a warning on the column names that are not in snake case format.

`isSnakeCase` *Used to check if a string is in snake case*

Description

This function is used check if a string conforms to the snake case format.

Usage

```
isSnakeCase(x)
```

Arguments

`x`	The string to evaluate

Value

TRUE if the string is in snake case

`isTaskRequired` *Is a task required when running in incremental mode*

Description

This function will attempt to check the `recordKeepingFile` to determine if an individual operation has completed by comparing the keys passed into the function with the checksum supplied

Usage

```
isTaskRequired(..., checksum, recordKeepingFile, verbose = TRUE)
```

Arguments

`...`	Parameter values used to identify the key in the incremental record keeping file
`checksum`	The checksum representing the operation to check
`recordKeepingFile`	A file path to a CSV file containing the record keeping information.
`verbose`	When TRUE, this function will output if a particular operation has completed based on inspecting the recordKeepingFile.

Value

Returns TRUE if the operation has completed according to the contents of the record keeping file.

`readCsv` *Used to read a .csv file*

Description

This function is used to centralize the function for reading .csv files across the HADES ecosystem. This function will automatically convert from snake_case in the file to camelCase in the data.frame returned as is the standard described in: https://ohdsi.github.io/Hades/codeStyle.html#Interfacing_between_R_and_SQL

Usage

```
readCsv(file, warnOnCaseMismatch = TRUE)
```

Arguments

`file`	The .csv file to read.
`warnOnCaseMismatch`	When TRUE, raise a warning if column headings in the .csv are not in snake_case format

Value

A tibble with the .csv contents

recordTasksDone *Record a task as complete*

Description

This function will record a task as completed in the `recordKeepingFile`

Usage

```
recordTasksDone(..., checksum, recordKeepingFile, incremental = TRUE)
```

Arguments

`...`	Parameter values used to identify the key in the incremental record keeping file
`checksum`	The checksum representing the operation to check
`recordKeepingFile`	A file path to a CSV file containing the record keeping information.
`incremental`	When TRUE, this function will record tasks otherwise it will return without attempting to perform any action

saveCohortDefinitionSet *Save the cohort definition set to the file system*

Description

This function saves a cohortDefinitionSet to the file system and provides options for specifying where to write the individual elements: the settings file will contain the cohort information as a CSV specified by the settingsFileName, the cohort JSON is written to the jsonFolder and the SQL is written to the sqlFolder. We also provide a way to specify the json/sql file name format using the cohortFileNameFormat and cohortFileNameValue parameters.

Usage

```
saveCohortDefinitionSet(
  cohortDefinitionSet,
  settingsFileName = "inst/Cohorts.csv",
  jsonFolder = "inst/cohorts",
  sqlFolder = "inst/sql/sql_server",
  cohortFileNameFormat = "%s",
  cohortFileNameValue = c("cohortId"),
  verbose = FALSE
)
```

Arguments

`cohortDefinitionSet`	The `cohortDefinitionSet` argument must be a data frame with the following columns: **cohortId** The unique integer identifier of the cohort **cohortName** The cohort's name **sql** The OHDSI-SQL used to generate the cohort Optionally, this data frame may contain: **json** The Circe JSON representation of the cohort
`settingsFileName`	The name of the CSV file that will hold the cohort information including the cohortId and cohortName
`jsonFolder`	The name of the folder that will hold the JSON representation of the cohort if it is available in the cohortDefinitionSet
`sqlFolder`	The name of the folder that will hold the SQL representation of the cohort.
`cohortFileNameFormat`	Defines the format string for naming the cohort JSON and SQL files. The format string follows the standard defined in the base sprintf function.
`cohortFileNameValue`	Defines the columns in the cohortDefinitionSet to use in conjunction with the cohortFileNameFormat parameter.
`verbose`	When TRUE, logging messages are emitted to indicate export progress.

`saveIncremental` *Used in incremental mode to save values to a file*

Description

When running in incremental mode, we may need to update results in a CSV file. This function will replace the `data` in `fileName` based on the key parameters

Usage

```
saveIncremental(data, fileName, ...)
```

Arguments

`data`	The data to record in the file
`fileName`	A CSV holding results in the same structure as the data parameter
`...`	Parameter values used to identify the key in the results file

`writeCsv` *Used to write a .csv file*

Description

This function is used to centralize the function for writing .csv files across the HADES ecosystem. This function will automatically convert from camelCase in the data.frame to snake_case column names in the resulting .csv file as is the standard described in: https://ohdsi.github.io/Hades/codeStyle.html#Interfacing_b

This function may also raise warnings if the data is stored in a format that will not work with the HADES standard for uploading to a results database. Specifically file names should be in snake_case format, all column headings are in snake_case format and where possible the file name should not be plural. See `isFormattedForDatabaseUpload` for a helper function to check a data.frame for rules on the column names

Usage

```
writeCsv(x, file, warnOnCaseMismatch = TRUE, warnOnUploadRuleViolations = TRUE)
```

Arguments

`x`	A data frame or tibble to write to disk.
`file`	The .csv file to write.
`warnOnCaseMismatch`	When TRUE, raise a warning if columns in the data.frame are NOT in camelCase format.
`warnOnUploadRuleViolations`	When TRUE, this function will provide warning messages that may indicate if the data is stored in a format in the .csv that may cause problems when uploading to a database.

Value

Returns the input x invisibly.

Index