Package 'Strategus'

October 4, 2023

Type Package

Title Coordinating and Executing Analytics Using HADES Modules

Version 0.1.0

Date 2023-10-04

Maintainer Anthony Sena <sena@ohdsi.org>

Description An R package for coordinating and executing analytics using HADES modules.

License Apache License 2.0

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

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

Depends R (>= 4.2.0),
CohortGenerator (>= 0.8.0),
DatabaseConnector (>= 6.2.3)

Imports targets,
renv (>= 1.0.0),
ParallelLogger (>= 3.1.0),
dplyr,
checkmate,
keyring,
rlang,
utils,
R.utils,
digest,
methods,
tibble,
ResultModelManager (>= 0.3.0),
SqlRender (>= 1.11.0)

Suggests testthat (>= 3.0.0),
fs,
knitr,
rmarkdown,
Eunomia,
withr

Remotes ohdsi/CohortGenerator,
ohdsi/ResultModelManager,
ohdsi/Eunomia

VignetteBuilder knitr

NeedsCompilation no

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

Encoding UTF-8

Config/testthat/edition 3

R topics documented:

`addModuleSpecifications` *Add module specifications to analysis specifications*

Description

Add module specifications to analysis specifications

Usage

```
addModuleSpecifications(analysisSpecifications, moduleSpecifications)
```

Arguments

`analysisSpecifications`
: An object of type `AnalysisSpecifications` as created by `createEmptyAnalysisSpecificiations`

`moduleSpecifications`
: An object of type `ModuleSpecifications`.

Value

Returns the `analysisSpecifications` object with the module specifications added.

addSharedResources *Add shared resources to analysis specifications*

Description

Add shared resources to analysis specifications

Usage

```
addSharedResources(analysisSpecifications, sharedResources)
```

Arguments

analysisSpecifications	An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations
sharedResources	An object of type SharedResources.

Value

Returns the analysisSpecifications object with the module specifications added.

createCdmExecutionSettings *Create CDM execution settings*

Description

Create CDM execution settings

Usage

```
createCdmExecutionSettings(
  connectionDetailsReference,
  workDatabaseSchema,
  cdmDatabaseSchema,
  cohortTableNames = CohortGenerator::getCohortTableNames(cohortTable = "cohort"),
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  workFolder,
  resultsFolder,
  minCellCount = 5,
  integerAsNumeric = getOption("databaseConnectorIntegerAsNumeric", default = TRUE),
  integer64AsNumeric = getOption("databaseConnectorInteger64AsNumeric", default = TRUE),
  resultsConnectionDetailsReference = NULL,
  resultsDatabaseSchema = NULL
)
```

Arguments

`connectionDetailsReference`
: A string that can be used to retrieve database connection details from a secure local store.

`workDatabaseSchema`
: A database schema where intermediate data can be stored. The user (as identified in the connection details) will need to have write access to this database schema.

`cdmDatabaseSchema`
: The database schema containing the data in CDM format. The user (as identified in the connection details) will need to have read access to this database schema.

`cohortTableNames`
: An object identifying the various cohort table names that will be created in the `workDatabaseSchema`. This object can be created using the `CohortGenerator::getCohortTableNa` function.

`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.

`workFolder`
: A folder in the local file system where intermediate results can be written.

`resultsFolder`
: A folder in the local file system where the module output will be written.

`minCellCount`
: The minimum number of subjects contributing to a count before it can be included in results.

`integerAsNumeric`
: Logical: should 32-bit integers be converted to numeric (double) values? If FALSE 32-bit integers will be represented using R's native `Integer` class. Default is TRUE

`integer64AsNumeric`
: Logical: should 64-bit integers be converted to numeric (double) values? If FALSE 64-bit integers will be represented using `bit64::integer64`. Default is TRUE

`resultsConnectionDetailsReference`
: A string that can be used to retrieve the results database connection details from a secure local store.

`resultsDatabaseSchema`
: A schema where the results tables are stored

Value

An object of type `ExecutionSettings`.

`createEmptyAnalysisSpecificiations` *Create an empty analysis specifications object.*

Description

Create an empty analysis specifications object.

Usage

```
createEmptyAnalysisSpecificiations()
```

Value

An object of type `AnalysisSpecifications`.

`createResultDataModels` *Create Result Data Models*

Description

Use this at the study design stage to create data models for modules This functions loads modules and executes any custom code to create schemas in a results database If recreate is set to TRUE all existing data will be removed, otherwise

Usage

```
createResultDataModels(
  analysisSpecifications,
  executionSettings,
  executionScriptFolder = NULL,
  keyringName = NULL,
  restart = FALSE
)
```

Arguments

`analysisSpecifications`	An object of type `AnalysisSpecifications` as created by `createEmptyAnalysisSpecificiations`
`executionSettings`	An object of type `ExecutionSettings` as created by `createCdmExecutionSettings()` or `createResultsExecutionSettings()`.
`executionScriptFolder`	Optional: the path to use for storing the execution script. when NULL, this function will use a temporary file location to create the script to execute.
`keyringName`	The name of the keyring to operate on. This function assumes you have created the keyring before calling this function. It defaults to NULL to select the default keyring. If the keyring is password protected, the password must be stored in the environment variable STRATEGUS_KEYRING_PASSWORD so it is retrieved using the command Sys.getenv("STRATEGUS_KEYRING_PASSWORD")
`restart`	Restart run? Requires `executionScriptFolder` to be specified, and be the same as the `executionScriptFolder` used in the run to restart.

createResultsExecutionSettings
Create Results execution settings

Description

Create Results execution settings

Usage

```
createResultsExecutionSettings(
  resultsConnectionDetailsReference,
  resultsDatabaseSchema,
  workFolder,
  resultsFolder,
  minCellCount = 5,
 integerAsNumeric = getOption("databaseConnectorIntegerAsNumeric", default = TRUE),
 integer64AsNumeric = getOption("databaseConnectorInteger64AsNumeric", default = TRUE)
)
```

Arguments

resultsConnectionDetailsReference	A string that can be used to retrieve the results database connection details from a secure local store.
resultsDatabaseSchema	A schema where the results tables are stored
workFolder	A folder in the local file system where intermediate results can be written.
resultsFolder	A folder in the local file system where the module output will be written.
minCellCount	The minimum number of subjects contributing to a count before it can be included in results.
integerAsNumeric	Logical: should 32-bit integers be converted to numeric (double) values? If FALSE 32-bit integers will be represented using R's native Integer class. Default is TRUE
integer64AsNumeric	Logical: should 64-bit integers be converted to numeric (double) values? If FALSE 64-bit integers will be represented using bit64::integer64. Default is TRUE

Value

An object of type ExecutionSettings.

ensureAllModulesInstantiated *Ensure all modules are instantiated*

Description

Ensure that all modules referenced in the analysis specifications are instantiated locally in the folder specified in the `INSTANTIATED_MODULES_FOLDER` environmental variable.

Missing modules will be fetched from remote repositories.

This function will also check whether there are different versions of the same module specified, which is not allowed, and whether all modules required by the specified modules are also instantiated.

Usage

```
ensureAllModulesInstantiated(analysisSpecifications)
```

Arguments

`analysisSpecifications`
: An object of type `AnalysisSpecifications` as created by `createEmptyAnalysisSpecificiations`

Value

A tibble listing the instantiated modules.

execute *Execute analysis specifications.*

Description

Execute analysis specifications.

Usage

```
execute(
  analysisSpecifications,
  executionSettings,
  executionScriptFolder = NULL,
  keyringName = NULL,
  restart = FALSE
)
```

Arguments

`analysisSpecifications`	An object of type `AnalysisSpecifications` as created by `createEmptyAnalysisSpecificiations`
`executionSettings`	An object of type `ExecutionSettings` as created by `createCdmExecutionSettings()` or `createResultsExecutionSettings()`.
`executionScriptFolder`	Optional: the path to use for storing the execution script. when NULL, this function will use a temporary file location to create the script to execute.
`keyringName`	The name of the keyring to operate on. This function assumes you have created the keyring before calling this function. It defaults to NULL to select the default keyring. If the keyring is password protected, the password must be stored in the environment variable STRATEGUS_KEYRING_PASSWORD so it is retrieved using the command Sys.getenv("STRATEGUS_KEYRING_PASSWORD")
`restart`	Restart run? Requires `executionScriptFolder` to be specified, and be the same as the `executionScriptFolder` used in the run to restart.

Value

Does not return anything. Is called for the side-effect of executing the specified analyses.

`getModuleList` *Provides a list of HADES modules to run through Strategus*

Description

This function provides a list of modules and their locations that may be used with Strategus.

Usage

```
getModuleList()
```

Value

A data.frame() of modules that work with Strategus. This will contain: module = The name of the module version = The version of the module remote_repo = The remote location of the module (i.e. github.com) remote_username = The organization of the module (i.e. OHDSI) module_type = 'cdm' or 'results'. 'cdm' refers to modules that are designed to work against patient level data in the OMOP CDM format. 'results' refers to modules that are designed to work against a results database containing output from a 'cdm' module.

retrieveConnectionDetails *Retrieve connection details from the secure location*

Description

Retrieve connection details from the secure location

Usage

```
retrieveConnectionDetails(connectionDetailsReference, keyringName = NULL)
```

Arguments

connectionDetailsReference	A string that can be used to retrieve the settings from the secure store.
keyringName	The name of the keyring to operate on. This function assumes you have created the keyring before calling this function. It defaults to NULL to select the default keyring. If the keyring is password protected, the password must be stored in the environment variable STRATEGUS_KEYRING_PASSWORD so it is retrieved using the command Sys.getenv("STRATEGUS_KEYRING_PASSWORD")

Value

Returns an object of type connectionDetails.

See Also

storeConnectionDetails()

runSchemaCreation *Create module(s) result data model*

Description

This function will create the results data model for the modules in the analysisSpecifications. A module can implement its own results data model creation function by implementing the function createDataModelSchema in its Main.R. The default behavior is to use the ResultsModelManager to create the results data model based on the resultsDataModelSpecification.csv in the module's results folder.

Usage

```
runSchemaCreation(
  analysisSpecifications,
  keyringSettings,
  moduleIndex,
  executionSettings,
  ...
)
```

Arguments

analysisSpecifications	An object of type AnalysisSpecifications as created by createEmptyAnalysisSpecificiations
keyringSettings	The keyringSettings from the executionSettings context
moduleIndex	The index of the module in the analysis specification
executionSettings	An object of type ExecutionSettings as created by createCdmExecutionSettings() or createResultsExecutionSettings().
...	For future expansion

---

storeConnectionDetails

Store connection details in a secure location

Description

Store connection details in a secure location

Usage

```
storeConnectionDetails(
  connectionDetails,
  connectionDetailsReference,
  keyringName = NULL
)
```

Arguments

connectionDetails	An object of type connectionDetails as created by the DatabaseConnector::createConnectionD function.
connectionDetailsReference	A string that can be used to retrieve the settings from the secure store.
keyringName	The name of the keyring to operate on. This function assumes you have created the keyring before calling this function. It defaults to NULL to select the default keyring. If the keyring is password protected, the password must be stored in the environment variable STRATEGUS_KEYRING_PASSWORD so it is retrieved using the command Sys.getenv("STRATEGUS_KEYRING_PASSWORD")

Value

Does not return anything. Is called for the side effect of having the connection details stored.

See Also

retrieveConnectionDetails()

unlockKeyring *Helper function to unlock a keyring*

Description

This helper function is used to unlock a keyring by using the password stored in Sys.getenv("STRATEGUS_KEYRING_P
It will alert the user if the environment variable with the password is not set.

Usage

```
unlockKeyring(keyringName)
```

Arguments

keyringName	The name of the keyring to operate on. This function assumes you have created the keyring before calling this function. It defaults to NULL to select the default keyring. If the keyring is password protected, the password must be stored in the environment variable STRATEGUS_KEYRING_PASSWORD so it is retrieved using the command Sys.getenv("STRATEGUS_KEYRING_PASSWORD")

Value

Returns TRUE if the keyring was unlocked using the password otherwise it returns FALSE

withModuleRenv *Load module execution space inside and renv inspired by targets::tar_script but allowing custom variable execution*

Description

Designed to allow more human readable code that is executed inside a module as well as simple variable substituion for injecting constants (e.g. simple parameters or file paths used inside and outside of modules)

Usage

```
withModuleRenv(
  code,
  moduleFolder,
  injectVars = list(),
  tempScriptFile = tempfile(fileext = ".R"),
  useLocalStrategusLibrary = TRUE,
  job = FALSE,
  processName = paste(moduleFolder, "_renv_run")
)
```

Arguments

code	code block to execute
moduleFolder	Instantiated Strategus module folder
injectVars	list of var names list(name=value) to replace (e.g. replace list(foo = "some string") will find the pattern foo and replace it with the string some string - be careful!
tempScriptFile	tempFile to write script to
useLocalStrategusLibrary	Use the locally installed Strategus library? TRUE will use the Strategus installation from the calling R process.
job	run as rstudio job
processName	String name for process

Details

This pattern also allows dependency injection which could be used if you don't want to use and renv and (instead) would like to use docker images or just execution in the base environment for testing/debugging

Value

NULL invisibly

Index