Creating Strategus Modules

Anthony G. Sena

2023-09-25

Contents

1 Background

This document aims to document the steps necessary to create analytic module that is compatible with Strategus. Please treat this document as a **work in progress** as Strategus is still under development.

2 Getting Started

A Strategus analytic module is an R Project that uses renv. **NOTE**: Please make sure you are using renv $>$ 1.0.0 when creating a new analytic module to make sure it is compatible with Strategus.

A Strategus module will contain the following files:

```
module_skeleton
+-- extras
|   +-- CreateJobContextForTesting.R
|   \-- ModuleMaintenance.R
```

```
+-- Main.R
+-- MetaData.json
+-- NEWS.md
+-- README.md
+-- renv
|   +-- activate.R
|   +-- settings.dcf
|   \-- settings.json
+-- renv.lock
+-- resources
|   +-- exampleAnalysisSpecifications.json
|   \-- resultsDataModelSpecification.csv
+-- SettingsFunctions.R
+-- tests
|   +-- test-eunomia.R
|   +-- testJobContext.rds
|   \-- testScript.R
\-- YourProjectModule.Rproj
```

Here we will detail how each file is used by Strategus and what is required in the contents of the file.

2.1 Creating YourProjectModule.Rproj and activating `renv`

This is the R Project (.Rproj) file for your module and should end in "Module". You may create this as a standard R Project via RStudio. Once the project is created, please use `renv::init()` to set up the `renv` folder as shown above. This will create the necessary `.RProfile` in the root of your project and the `renv` subfolder with the necessary R code for `renv`'s operations.

2.2 README.md

This is a standard README markdown file that describes the module.

2.3 NEWS.md

This is a standard NEWS markdown file that is a change log for your module. See this post for more information.

2.4 MetaData.json

MetaData.json holds information that describes the module and its dependencies:

```
{
  "Name": "CohortGeneratorModule",
  "Version": "0.0.1",
  "Dependencies": ["SomePrerequisiteModule"],
  "TablePrefix": "cg_"
}
```

To detail the contents of the JSON file:

- **Name**: The name of the module
- **Version**: The version of the module. This should have a corresponding git tag in the repository when the module is released otherwise Strategus will not be able to download it.
- **Dependencies**: A list of modules that are required to have successfully executed **BEFORE** this module is executed. If there are no dependencies, leave this as an empty array `[]`.
- **TablePrefix**: The prefix to use when creating the results tables in the `resultsDataModelSpecification.csv`. Please see Main.R for more information on how this value is used.

2.5 Main.R

This file holds the main executable for your module. This file must contain a function called `execute(jobContext)`.

```
execute <- function(jobContext) {
  # VALIDATE THE jobContext
  rlang::inform("Validating inputs")

  # YOUR VALIDATION CODE GOES HERE....

  # EXECUTE THE ANALYTICS
  rlang::inform("Executing")

  # YOUR EXECUTION CODE GOES HERE....

  # ASSEMBLE AND .ZIP THE RESULTS
  rlang::inform("Export data")

  # YOUR CODE GOES HERE....
}
```

As shown in the code above, your `execute(jobContext)` should handle: validating the `jobContext` object to ensure it has all of the information necessary for your code to function, a section to execute the analytics and finally code to assemble the output. Here we will describe the requirements for the way in which your module must output its results:

- A single .ZIP file is created that holds all of the result files as described below.
- Output files are required to be in .CSV format. Use CohortGenerator v0.5.0 or higher which contains a helper function for `writeCsv()` which will ensure your output is formatted properly. For more information, please see: https://ohdsi.github.io/CohortGenerator/reference/writeCsv.html. **IMPORTANT:** File names *must* correspond to the table names that are specified in the `resultsModuleSpecification.csv`.
- You must include a file named `resultsModuleSpecification.csv` in your output directory. The format of this file is as follows:

```
table_name,column_name,data_type,is_required,primary_key,empty_is_na
my_table,cohort_id,bigint,Yes,Yes,No
my_table,cohort_name,varchar,Yes,No,No
my_table,generation_status,varchar,No,No,No
my_table,start_time,Timestamp,No,No,No
my_table,end_time,Timestamp,No,No,No
my_table,database_id,varchar,Yes,Yes,No
```

The `resultsModuleSpecification.csv` has the following columns:

- **table_name**: The table name to use to hold the data.
- **column_name**: The column name in the table.
- **data_type**: The data type for the column. See https://www.postgresql.org/docs/current/datatype.html for examples.
- **is_required**: Will this column allow for NULL values? Yes/No
- **primary_key**: Is this column part of the table's primary key? Yes/No

2.6 SettingsFunctions.R

This file contains one or more functions required to create the module settings for use in Strategus. We plan to later remove this requirement when we can describe the module specification using the OpenAPI 3.0 Specification. For now, your module should contain a function similar to the following:

```
createCohortGeneratorModuleSpecifications <- function(incremental = TRUE,
                                                       generateStats = TRUE) {
  analysis <- list()
  for (name in names(formals(createCohortGeneratorModuleSpecifications))) {
    analysis[[name]] <- get(name)
  }

  checkmate::assert_file_exists("MetaData.json")
  moduleInfo <- ParallelLogger::loadSettingsFromJson("MetaData.json")

  specifications <- list(module = moduleInfo$Name,
                         version = moduleInfo$Version,
                         remoteRepo = "github.com",
                         remoteUsername = "ohdsi",
                         settings = analysis)
  class(specifications) <- c("CohortGeneratorModuleSpecifications", "ModuleSpecifications")
  return(specifications)
}
```

As shown above, this example comes from the `CohortGeneratorModule` and the function name reflects the fact that the function will create the settings used to dictate the behavior of the module. The parameters of the function will differ based on the requirements of your module - if there are choices to be made when running your module, you should include these as parameters to your module specification function.

Internal to the function above, the formal parameters to the function are used to construct a `list()` named `analysis` which holds the analysis settings. Next the `MetaData.json` file is used to obtain the module name/version for inclusion in the `specifications` list. The `specifications` list contains the `remoteRepo` and `remoteUsername` properties to indicate where your module is stored on GitHub. Finally, we set the `class()` of the `specifications` object to `c("CohortGeneratorModuleSpecifications", "ModuleSpecifications")`. For your module, you will want to substitute `"CohortGeneratorModuleSpecifications"` for the name of your module and retain the `"ModuleSpecifications"` in the vector.

The following JSON fragment shows how the output of `createCohortGeneratorModuleSpecifications()` is used in the `moduleSpecifications` section of the overall analysis settings JSON for Strategus:

```
{
  "sharedResources": [
    {
      "cohortDefinitions": [
        {
```

```
          "cohortId": "1",
          "cohortName": "celecoxib",
          "cohortDefinition": "...truncated..."
        }
      ],
      "attr_class": ["CohortDefinitionSharedResources", "SharedResources"]
    }
  ],
  "moduleSpecifications": [
    {
      "module": "CohortGeneratorModule",
      "version": "0.0.1",
      "remoteRepo": "github.com",
      "remoteUsername": "ohdsi",
      "settings": {
        "incremental": true,
        "generateStats": true
      },
      "attr_class": ["CohortGeneratorModuleSpecifications", "ModuleSpecifications"]
    }
  ],
  "attr_class": "AnalysisSpecifications"
}
```


2.7 renv.lock

Each module will make use of renv to capture its R package dependencies. Furthermore, Strategus will make use of the `renv` settings in your module to create a run-time environment when executing your module to ensure all of the necessary dependencies are available to your module.

It is recommended to use the HADES-wide renv.lock file which is available at https://github.com/OHDSI/Hades/blob/main/hadesWideReleases. Find the most recent release based on the folder name and copy the `renv.lock` file into the root of your module project.

If you need to install additional dependencies for your project, use `renv::record()` to record it in the lock file.

3 Extra files

It is advisable to add an `extras` folder to your project to include other useful files for managing and testing your module. We'll detail those files here:

3.1 ModuleMaintenance.R

This file is used to store utility functions for your module, such as the code mentioned earlier for generating the renv.lock file. Here is an example of the contents of ModuleMaintenance.R:

```
# Format and check code:
styler::style_dir()
OhdsiRTools::updateCopyrightYearFolder()
OhdsiRTools::findNonAsciiStringsInFolder()
devtools::spell_check()
```

```
# Generate renv lock file and activate renv:
OhdsiRTools::createRenvLockFile(
  rootPackage = "CohortGenerator",
  includeRootPackage = TRUE,
  mode = "description",
  additionalRequiredPackages = c("checkmate", "CirceR")
)
renv::init()
```

3.2 Test Files

The following file is used to create a test jobContext for use in the `execute(jobContext)` as described in Main.R:

https://github.com/OHDSI/CohortGeneratorModule/blob/main/extras/test/CreateJobContext.R

And the following file is used to create a test harness for running your module:

https://github.com/OHDSI/CohortGeneratorModule/blob/main/extras/test/TestModuleStandalone.R